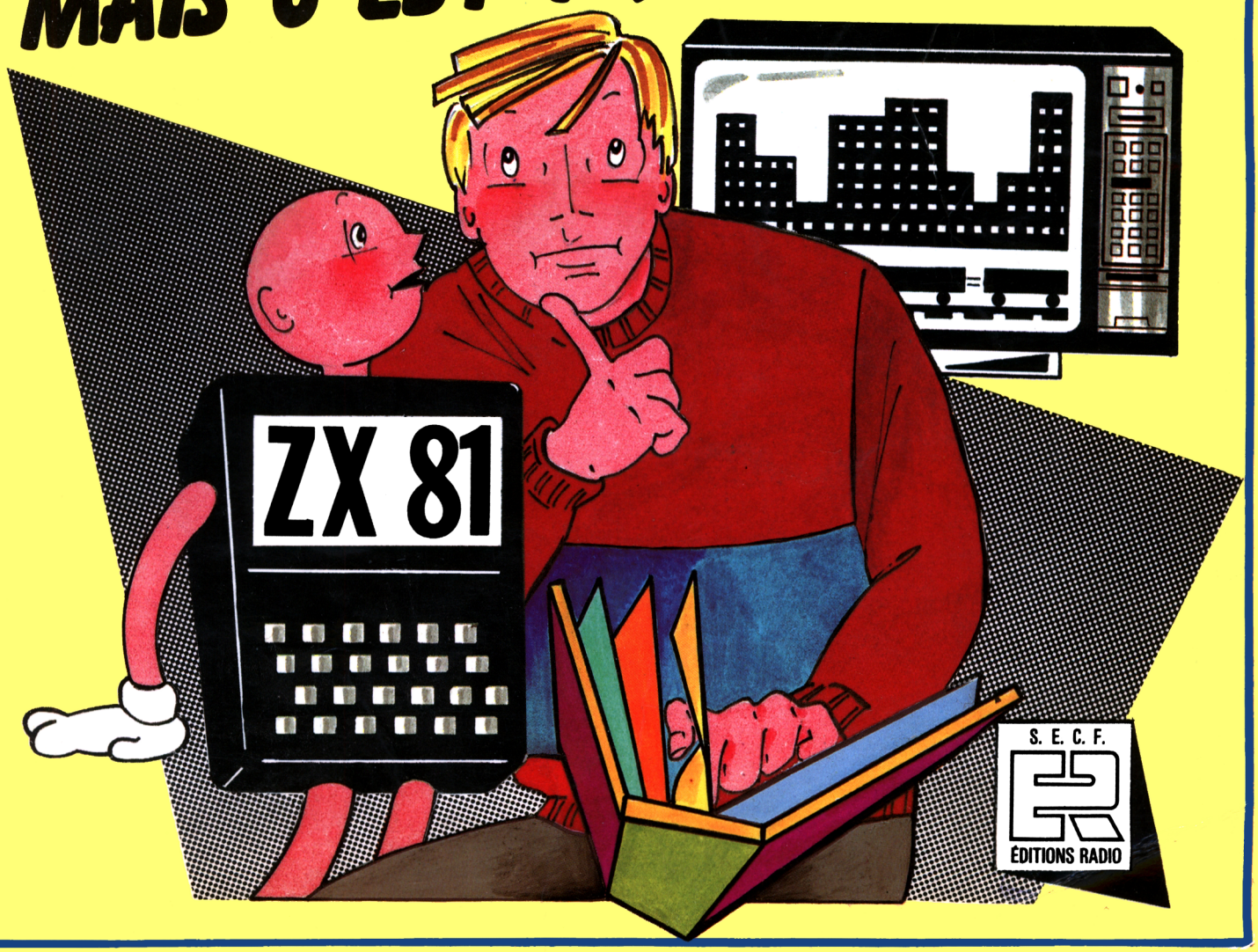
J.C. FANTOU

CRÉER ANIMER ses JEUX VIDÉO

MAIS C'EST TRÈS SIMPLE...



S. E. C. F.
ÉDITIONS RADIO

S. E. C. F.

Editions Radio

9, RUE JACOB - 75006 PARIS
TEL. 329.63.70

J.C. FANTOU

CRÉER ANIMER ses JEUX VIDÉO sur ZX 81

Dessins marginaux de Françoise Jouvion

S. E. C. F.

Editions Radio

9, RUE JACOB - 75006 PARIS

TEL. 329.63.70

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal.

© SECF Éditions Radio, Paris
1984

Tous droits de traduction, de reproduction et d'adaptation réservés pour tous pays

Imprimerie Berger-Levrault, Nancy

Dépôt légal : février 1984
Éditeur n° 949 - Imprimeur : 778683
I.S.B.N. 2 7091 0937 9

A mon fils Olivier sans qui
cet ouvrage n'aurait jamais existé.

AVANT-PROPOS

Pourquoi avoir écrit un livre de plus sur les jeux vidéo, vous demandez-vous, alors qu'il en existe déjà des centaines sur le marché ?

A cette question, je répondrais par une petite histoire.

Un jour, mon fils Olivier qui vient d'avoir 14 ans, est revenu tout excité du lycée en m'annonçant :

— *Papa, un club d'informatique vient de s'ouvrir à l'école ! Est-ce que je peux m'y inscrire ?*

— *Bien sûr,* répondis-je, *cela sera très enrichissant pour toi d'apprendre l'informatique, car à notre époque...bla...bla...bla...bla...*

J'ai bien dû disserter pendant au moins une demi-heure sur le sujet, approuvant fortement cette heureuse initiative et le faisant savoir à qui voulait bien tendre une oreille complaisante.

La première séance d'informatique devait avoir lieu le matin du samedi suivant. Je le vis donc partir, heureux et enthousiaste vers son lycée, situation assez inhabituelle il est vrai, si l'on songe qu'il y passe déjà cinq journées entières par semaine. A son retour, je vis à sa mine déconfite que quelque chose n'allait pas. Je le questionnai :

— *Alors ! Comment c'était ?*

— *La-men-ta-ble !* me répondit-il.

Toute une réponse condensée en un seul mot. Que certains mots peuvent peser lourd dans la bouche d'un enfant !

Je voulais des précisions :

— *Que s'est-il passé ?*

— *L'animateur nous a dit que nous allions étudier le BASIC et que grâce à ce langage, nous allions pouvoir effectuer des opérations mathématiques sur un micro-ordinateur.*

— *Et le dessin à l'écran,* lui demandais-je ?

— *Il a dit que d'ici quelques mois, nous pourrions dessiner quelques figures géométriques simples.*

— *Et alors,* lui dis-je, *où est le problème ?*

— *Le problème,* me répondit-il, *c'est que cela ne m'intéresse pas du tout d'aller le samedi au lycée pour faire du calcul et de la géométrie. Si je veux apprendre l'informatique, ce n'est pas pour faire avec un ordinateur ce que je sais faire avec une calculatrice ou un crayon et une règle. Non, moi, ce qui m'intéresse, c'est de pouvoir inventer des jeux vidéo. Et sur ce point, l'animateur a été formel. Il nous a déclaré : « Pas question de faire du jeu vidéo ici ; ce n'est pas sérieux et de plus, ce n'est pas au programme ! »*

Tout compte fait, je trouvais que les réflexions d'Olivier n'étaient pas idiotes du tout. En effet, à quoi cela pouvait-il servir de vouloir promouvoir l'informatique à travers des exemples qui ne faisaient qu'en démontrer l'inutilité. Par contre, l'univers des jeux vidéo étant en constante expansion, pourquoi ne pas se servir de l'engouement qu'il suscite auprès des jeunes pour promouvoir efficacement l'informatique ?

Fort de mes conclusions, je lui conseillais :

— *Pourquoi n'achèterais-tu pas un livre sur les jeux vidéo et n'apprendais-tu pas tout seul comment les réaliser ?*

Cette idée le séduisit tant et si bien, qu'il passat l'après-midi entière à fouiner dans plusieurs librairies plus ou moins spécialisées en informatique, dans le but évident de ramener un livre sur l'art et la manière de dessiner un jeu vidéo sur le téléviseur du salon, en se servant d'un micro-ordinateur domestique.

Je le vis revenir en fin de soirée, harassé, bougon, affichant sur son visage cet air de profond découragement qui en disait long sur les résultats de ses recherches. Je le questionnais néanmoins :

— *Alors, tu as trouvé quelque chose ?*

— *Tu sais,* me répondit-il, *des livres sur l'informatique, il y en a des tonnes !* (là, il exagérait un peu !). *J'en ai trouvé deux ou trois qui donnent des programmes de jeux tout faits, mais je n'y comprends rien ! Quant à trouver* **un livre qui explique** *comment créer et animer ses propres jeux vidéo, alors-là, rien !*

Et il ajouta aussitôt :

— *Dis donc Papa,* toi qui écris des livres sur l'électronique, *pourquoi tu n'en écrirais pas un sur la façon de concevoir soi-même ses propres jeux vidéo ?*

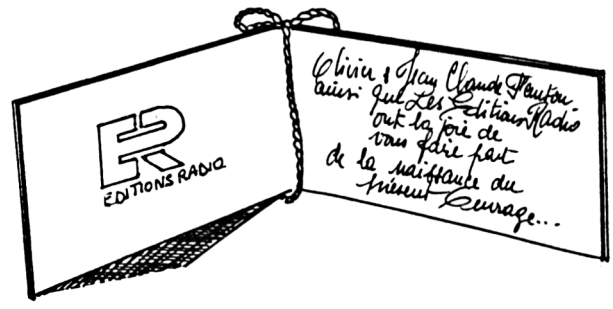
Vous imaginez la réponse que j'ai pu lui faire, surtout j'étais persuadé que ce qu'il recherchait devait exister quelque part et qu'il ne s'était tout simplement pas adressé au bon endroit. Me voilà donc parti la semaine suivante, en quête d'un ouvrage d'informatique sur les jeux vidéo. Les librairies spécialisées ou non, les boutiques d'informatique, bref, tout y est passé ! Résultat, rien ! Pas un seul ouvrage qui explique comment élaborer un jeu vidéo du début jusqu'à la fin.

Il ne me restait plus qu'à m'exécuter et à écrire ce fameux livre, car j'imaginais que mon fils ne devait pas être le seul à ressentir ainsi du découragement, face à un domaine qui aurait dû soulever l'enthousiasme.

Il a donc fallu tout d'abord investir dans l'achat d'un micro-ordinateur. Notre choix s'est tout de suite porté sur le moins cher du marché, à savoir le modèle ZX 81 de SINCLAIR. Ce petit ordinateur, n'est en fait petit que par la taille et peut rivaliser avec bien des plus gros. De surcroît, il utilise le téléviseur familial pour visualiser les graphismes vidéo qu'on lui fait exécuter, ce qui économise l'achat d'une « visu ».

Restait ensuite à définir une méthode de travail. Elle émergea très vite de nos premiers tatonnements : Olivier me disait que ce qu'il souhaitait obtenir, et moi, je me creusais la tête pour essayer de trouver une solution ! Dans le fond, c'était lui la tête et moi le bras.

C'est ainsi qu'est né le présent ouvrage « Créer et animer ses jeux vidéo », fruit d'une collaboration étroite entre le fils qui voulait apprendre l'informatique à sa façon, et le père qui possèdait un fragment de la connaissance.



COMMENT DESSINER SUR UN ECRAN

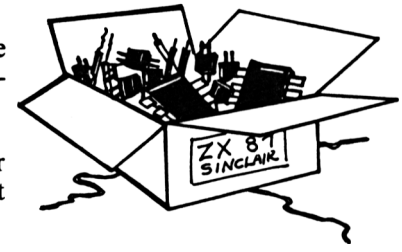
La mise en route du micro-ordinateur

Me voici donc, un soir du mois d'octobre, rentrant à la maison avec un paquet sous le bras. Olivier, qui était déjà installé à son bureau en train de faire ses devoirs, s'écria en me voyant arriver :

O. — Est-ce que c'est l'ordinateur que tu rapportes là ?

Le "là" correspondait à un doigt tendu vers le paquet.

P. — Oui, mais je te préviens, il est en pièces détachées. Il va falloir que tu me laisses le temps de le câbler.



O. — Ah ! (le ton sur lequel ce "Ah" était prononcé semblait indiquer que plus vite l'ordinateur passerait de l'état de pièces détachées à celui d'un ensemble en état de marche, mieux je serais considéré !).

Ainsi fut fait et tard dans la soirée, le dernier composant était soudé, le boîtier refermé et l'ensemble prêt à fonctionner (du moins je l'espérais !). Olivier, qui surveillait d'un œil l'état d'avancement des travaux s'écria aussitôt :

O. — Ça y est, tu as fini ! Je vais enfin pouvoir l'essayer.

Bien que le "enfin" me parut de trop, j'évitais de le lui faire remarquer et, consultant ma montre, je déclarais :

P. — Dis donc, tu as vu l'heure ? Il est vraiment trop tard pour l'essayer maintenant. Reportons cela à demain matin.

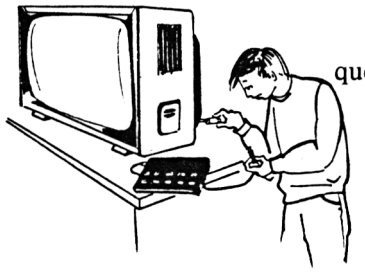
Autant dire que la nuit fut courte et peuplée de rêves particulièrement "informatisés". Le lendemain matin, j'eus droit à toute une série d'attentions très particulières, visant notamment à accélérer mon passage inévitable par la table du petit déjeuner. J'en eus d'ailleurs la confirmation lorsqu'il me déclara tout net :

O. — Bon, maintenant que tu as déjeuné, si on passait aux choses sérieuses ?

P. — Ok, je suis prêt. Par quoi commençons-nous ?

O. — J'aimerais que tu me montres comment il faut brancher le micro-ordinateur sur le téléviseur.

Heureusement que pour cela il suffisait de consulter la notice d'emploi livrée avec l'appareil et que sa lecture nous fournit les renseignements que nous recherchions. Après divers branchements et le réglage du téléviseur sur le canal 36, je dis à Olivier :

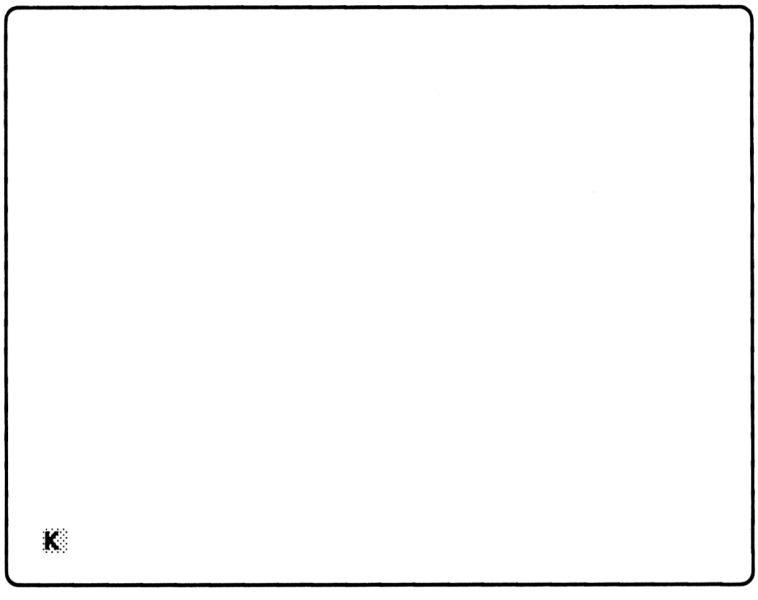


P. — Normalement, si je ne me suis pas trompé, tu devrais pouvoir faire apparaître quelque chose sur l'écran en tapant sur les touches du clavier.

O. — Mais il y a déjà une lettre inscrite dans le bas à gauche de l'écran.

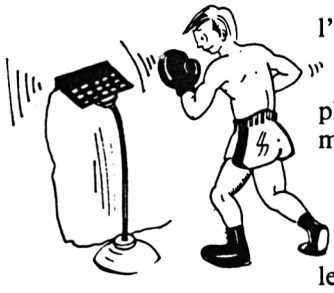
P. — Qu'est-ce que c'est comme lettre ?

O. — C'est un K dans un carré noir.



P. — Ça, c'est un curseur qui t'indique que l'ordinateur attend tes ordres. Tu verras également la lettre L apparaître à la place du K après avoir tapé sur une touche d'instruction.

Et c'est ce qui se passa. Après une suite de « schmilblics » imprononçables du style PLOT WERTYUIOP, ponctués d'un "Youppi, ça marche !", j'eus droit à ma première question :



O. — Bon, c'est pas tout cela, mais comment fait-on pour effacer ce qu'il y a sur l'écran ?

P. — Il te suffit d'appuyer en même temps sur la touche marquée [RUBOUT], placée en haut à droite du clavier, ainsi que sur la touche [SHIFT]. Tu repéreras facilement la touche [SHIFT] car elle est écrite en rouge.

O. — Ok, ça marche, mais cela n'efface qu'une lettre à la fois.

P. — C'est normal ! Tu dois appuyer autant de fois sur [RUBOUT] que tu as de lettres à effacer.

O. — Qu'est-ce que veut donc dire RUBOUT ?

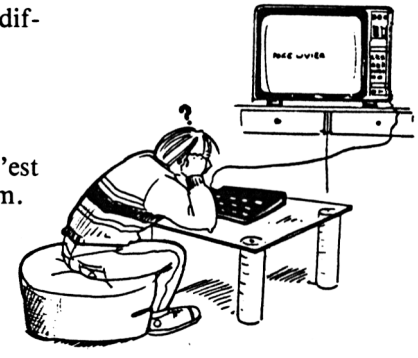
P. — Il s'agit d'une instruction en langage BASIC. C'est un langage très simple écrit pour ceux qui débutent en informatique. Comme ce sont des Américains qui l'ont inventé, toutes les instructions de ce langage sont écrites en anglais. « Rubout » est un mot qui veut dire « effacer ».

O. — Et SHIFT, qu'est-ce que cela veut dire ?

P. — Il s'agit de la touche qui, sur les machines à écrire, permet d'écrire des majuscules à la place des minuscules.

O. — Bien ! Maintenant que j'ai effacé les « cochonneries » que j'avais écrites, je vais essayer d'écrire mon prénom.

P. — Bonne idée ! (en réalité, je savais qu'Olivier allait se heurter à une nouvelle difficulté et je voulais qu'il en prenne conscience).



Effectivement, au bout d'un moment, il me posa la question suivante :

O. — Dis donc Papa, je ne sais si c'est moi qui ne sais pas m'en servir ou si c'est l'ordinateur qui est en panne, mais je n'arrive pas à écrire correctement mon prénom.

P. — Qu'est-ce que tu obtiens ?

O. — J'obtiens POKE LIVIER.

P. — C'est normal. Cela tient au fait que tu tapes sur un clavier d'ordinateur et non sur celui d'une machine à écrire. Sur un ordinateur, avant d'entrer des caractères, il faut lui dire ce qu'il doit en faire. Ainsi, si tu veux qu'il écrive ton prénom, indique-lui par une instruction que c'est cela que tu attends de lui.

O. — Quelle instruction dois-je alors utiliser pour écrire mon prénom ?

P. — L'instruction PRINT !

O. — Ah oui, c'est vrai, ''print'' en anglais, cela veut dire ''imprimer'' !

P. — Bien. Il ne te reste plus qu'à effacer ce que tu as écrit et à tout retaper après l'instruction PRINT.

O. — Je dois tout effacer ? Je ne peux pas simplement effacer le POKE inutile et rajouter ce qui manque !

P. — Si, bien sûr ! Tu n'as qu'à déplacer ton curseur à l'aide de la touche marquée ⇦ ou ⇨, en n'oubliant pas d'appuyer sur [SHIFT] et, lorsque le curseur sera juste à la droite de POKE, de l'effacer à l'aide de la touche [RUBOUT] .

O. — Voilà, c'est fait. Il ne me reste plus qu'à taper sur [PRINT] et à rajouter un O. Ça y est, j'ai réussi ! Mais au fait, pourquoi en appuyant sur la touche [P], j'ai obtenu l'instruction PRINT et non la lettre P, alors qu'en tapant sur la touche [O], j'ai obtenu la lettre O ?

P. — Tout simplement parce que l'ordinateur sait que la première chose que tu dois taper avant d'écrire quoi que ce soit, est une instruction. Il sélectionne donc de lui-même l'instruction qui est placée au-dessus de la touche lorsqu'il s'agit de la première touche sur laquelle tu appuies.

O. — Compris ! Dis donc, c'est astucieux ça comme truc ! Ainsi, quand je tape sur la touche [O], comme c'est la deuxième touche sur laquelle je frappe, je n'obtiens que la lettre O.

P. — Eh oui ! Tu vois comme c'est simple. Je vais même te montrer un truc. Quand le curseur indique la lettre K, c'est que l'ordinateur va sélectionner de lui-même l'instruction placée au-dessus de la touche. Sinon, c'est la lettre L qui est visible.

O. — Exact ! J'avais un K avant de taper PRINT et maintenant j'ai un L. Bien, mais tout ça ne me dit pas ce que je dois faire avec mon PRINT OLIVIER ?

P. — Il suffit que tu appuies sur la touche [NEW LINE] pour signaler à l'ordinateur que tu as fini d'écrire ta ligne d'instruction et que tu attends de lui qu'il l'exécute.

O. — Je viens d'appuyer sur [NEW LINE] et tout a disparu. Tout ce qu'il y a sur l'écran c'est deux chiffres !

P. — Qu'est-ce que c'est comme chiffres ?

```
2 / Ø
```

O. — C'est un 2/Ø ! Au fait, pourquoi le Ø est-il barré ?

P. — Pour ne pas le confondre avec la lettre O... Les deux chiffres que tu vois correspondent à un message d'erreur. C'est ainsi que l'ordinateur dialogue avec toi. Le chiffre 2 signale que tu n'as pas respecté la syntaxe d'écriture de l'instruction PRINT.

O. — Ah, bon ! Et qu'est-ce qui manque ?

P. — Des guillemets ! Il faut mettre entre guillemets ce que tu veux que l'ordinateur t'écrive.

O. — Comme ça ?

```
PRINT "OLIVIER"
```

P. — Oui !

O. — Et si j'appuie maintenant sur [NEW LINE], cela ne s'effacera pas comme tout à l'heure ?

P. — Normalement, non ! Essaye donc pour voir.

```
OLIVIER
```

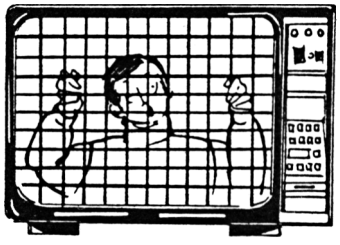
Et le prénom d'Olivier s'afficha enfin dans le haut de l'écran, ce qui, somme toute, ne sembla pas lui procurer le plaisir escompté. Il me déclara aussitôt :

O. — Tu va peut-être dire que je suis compliqué, mais j'aimerais pouvoir centrer mon nom dans le milieu de l'écran. Est-ce possible ?

P. — Bien sûr, mais pour cela il faut que je t'explique comment l'écran est quadrillé.

Le quadrillage de l'écran

P. — Apprends que l'écran d'affichage de ton micro-ordinateur comporte 22 lignes et 32 colonnes. Cela constitue une mosaïque de 704 petits carrés dans lesquels tu peux écrire ce que tu veux.



O. — Un peu comme à la bataille navale, mais en plus grand ?

P. — C'est ça !... Chaque ligne est numérotée de Ø à 21, la ligne Ø étant celle qui est placée tout en haut de l'écran et la ligne 21, tout en bas.

O. — Et je suppose que, pour les colonnes, la colonne Ø est celle qui est la plus proche du bord gauche de l'écran, tandis que, la colonne 31 est la plus proche du bord droit.

P. — Exact !... Je dois toutefois te signaler qu'il reste deux lignes supplémentaires situées entre la ligne 21 et le bas de l'écran, que l'ordinateur se réserve pour ses besoins personnels et auxquelles tu ne peux accéder.

O. — Bon, d'accord, mais tout cela ne me dit pas comment je peux inscrire mon prénom au milieu de l'écran.

P. — On y arrive !... Quand tu demandes à l'ordinateur de t'imprimer quelque chose sur l'écran, et que tu ne lui précises pas où tu veux qu'il te l'imprime, il va tout bêtement commencer à écrire sur la ligne du haut, comme tu le ferais sur une page neuve de cahier.

O. — Tiens, c'est vrai !

P. — Si, par contre, tu lui dis que tu veux qu'il commence à écrire à partir de la 11e ligne et que sur cette ligne, il ne démarre qu'à partir de la 11e colonne, il le fera !

O. — Et quelle est l'instruction qui me permet d'obtenir cela ?

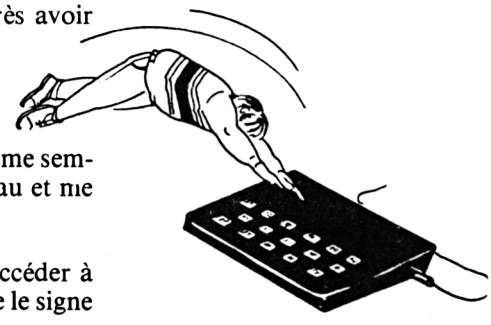
P. — C'est l'instruction PRINT AT, suivie du numéro de la ligne et de celui de la colonne.

O. — Ok, j'ai compris ! Je vais demander à l'ordinateur d'écrire mon prénom sur la 11e ligne, en débutant à la 11e colonne, comme tu me l'as suggéré.

Et Olivier se penche sur son clavier, bien décidé à découvrir tout seul la fameuse instruction PRINT AT. Au bout d'un moment, il émerge et me demande :

O. — Dis-moi Papa, je n'ai rien trouvé d'autre que l'instruction PRINT que j'ai déjà utilisée et l'instruction AT placée en dessous de la touche [C].

P. — C'est normal, car il s'agit en fait de deux instructions séparées. Après avoir tapé sur [PRINT], tu appuies ensuite sur [AT].



O. — Ah bon !...

Et il replonge la tête la première dans son clavier... Après quelques essais qui me semblèrent infructueux, vu ce qui apparaissait sur l'écran, je le vis émerger à nouveau et me déclarer :

O. — Décidément, ce n'est toujours pas évident ! Comment fais-tu pour accéder à l'instruction AT ? J'ai beau appuyer sur [SHIFT], je n'obtiens rien d'autre que le signe d'interrogation. Où est l'astuce ?

P. — L'astuce, comme tu dis, consiste à appuyer en même temps sur [SHIFT] et sur [FUNCTION], et ensuite sur [AT]. D'ailleurs, c'est comme cela que tu dois procéder chaque fois que tu désires obtenir une instruction placée en dessous d'une touche.

O. — C'est pas génial comme astuce !

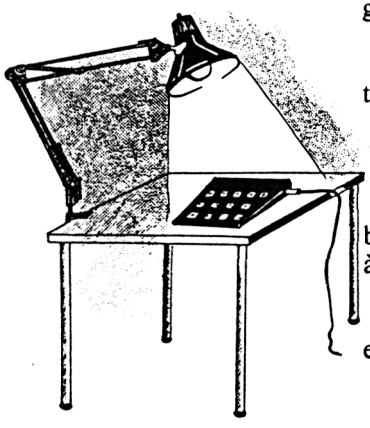
P. — Peut-être, mais cela n'a été inventé que pour économiser des touches de clavier.

O. — Ah bon !...

Et le voilà qui replonge dans son clavier. Décidément, il va falloir que j'améliore l'éclairage du bureau...

O. — Ça y est, j'ai réussi !... Après le AT, j'ai tapé deux fois le nombre 11, une fois pour désigner la ligne et une fois pour désigner la colonne. Ça doit être bon maintenant ?

P. — Presque !... Pour respecter la syntaxe de cette instruction, il faut taper une virgule entre les deux nombres et un point-virgule après le second nombre.



O. — Décidément, qu'est-ce que c'est compliqué !... Qu'est-ce qui se passe si je me trompe ou si j'oublie un signe ?

P. — Essaye donc, tu verras bien !

O. — D'accord !... J'ai fait exprès d'oublier le point-virgule à la fin du second nombre. J'appuie sur [NEWLINE]... Tiens, il vient d'apparaître une lettre S sur fond noir à côté du L du curseur.

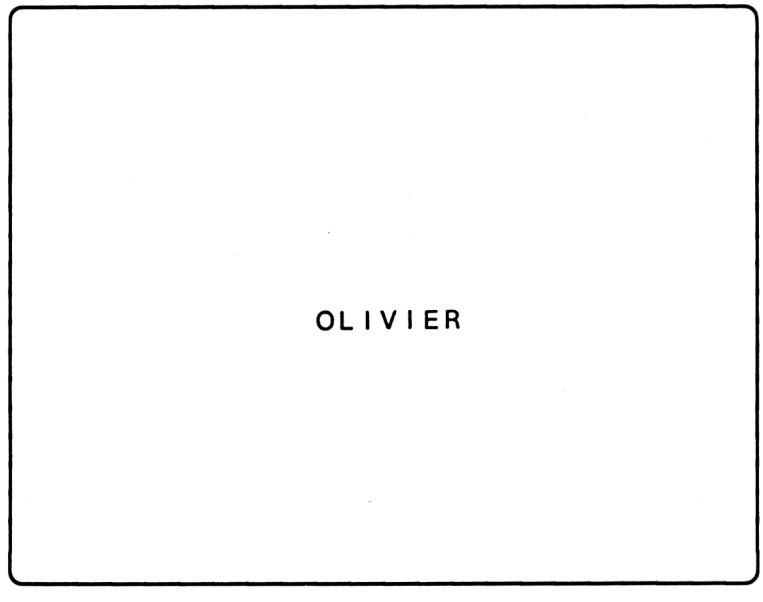
P. — Ce S est là pour t'indiquer que tu as fait une erreur de syntaxe, c'est-à-dire une erreur dans la façon exacte d'écrire cette instruction.

O. — Ah, je comprends ! S comme syntaxe !

P. — C'est ça !

O. — Bien !... Il ne me reste plus qu'à rajouter le point-virgule qui manque et cela devrait être bon...

```
PRINT AT 11,11;"OLIVIER"
```



P. — Et alors ?

O. — Ça y est !... Enfin !

Olivier est en extase devant le tour de force qu'il vient d'accomplir. Je le ramène à la réalité.

P. — Puique maintenant tu sais comment il faut faire, amuse-toi donc à écrire ce que tu veux et à le placer où bon te semble.

O. — Bonne idée !

Et je laisse Olivier en train de pianoter, bien décidé à occuper les quelques instants de répit qu'un tel exercice est censé m'octroyer. Je ne m'étais pas éloigné depuis deux minutes que déjà il me rappelait :

O. — Mayday !... Mayday !... (pour ceux qui ne connaîtraient pas, il s'agit d'un appel de détresse en usage sur certaines bandes dessinées où il est question d'aviateurs...)

P. — J'arrive !... Que se passe-t-il ?

O. — Je viens de réussir à bloquer l'ordinateur !

P. — En quoi faisant ?

O. — Rien de particulier !... J'ai tout simplement essayé d'écrire mon prénom à la droite de l'écran, sur la ligne du bas.

P. — Et que s'est-il passé ?

O. — L'ordinateur m'a affiché un O tout seul, après quoi il m'a adressé un ..., comme tu appelles cela ?...

P. — Un message d'erreur !

O. — Oui, c'est ça !... Un message d'erreur constitué par un 5/Ø. Qu'est-ce que cela veut dire ?

P. — Avant de te répondre, il me faudrait savoir comment tu as rédigé ton instruction.

O. — J'ai tout simplement écrit : PRINT AT 21, 31 ; "OLIVIER"

P. — Bien !... L'ordinateur t'a adressé ce message d'erreur tout bonnement parce que tu lui as demandé d'écrire quelque chose en-dehors de l'écran, et ça, il ne sait pas le faire.

O. — Pourtant, il y a bien 31 colonnes ?

P. — Oui, mais il n'y a que 21 lignes. Si tu lui avais demandé la même chose sur la ligne 11, il aurait tout bonnement écrit le O à la fin de cette ligne et la suite au début de la ligne suivante. Seulement là, tu le lui as demandé sur la dernière ligne. Il ne pouvait donc effectuer le report sur la ligne suivante et c'est pour cela qu'il t'a adressé ce message d'erreur.

O. — Si je comprends bien, il faut donc que je me débrouille pour que la dernière lettre de mon prénom tombe pile sur la 31e colonne, si je veux l'écrire tout à fait à droite, sans coupure ou report sur la ligne suivante ?

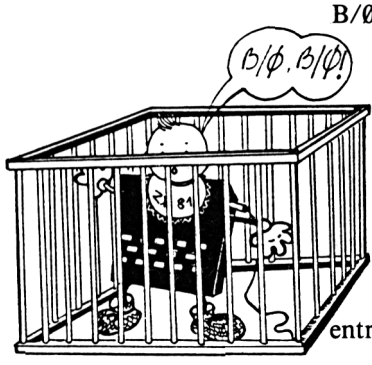
P. — C'est exact !

O. — Bon !... Attends ... Je calcule !... Comme mon prénom comporte 7 lettres, il suffit que je commence à l'écrire à la $31 - 7 = 24^e$ colonne.

P. — Pas tout à fait !... Moi, j'essayerais plutôt la 25e. N'oublie pas qu'il faut toujours rajouter une unité au résultat de ce type de soustraction.

O. — Ok !... Au fait, qu'est-ce qui se passe si j'écris un numéro de colonne qui n'existe pas, comme 5Ø par exemple ?

P. — Essaye donc, tu verras bien !... Normalement, il devrait te répondre 5/Ø ou B/Ø.



O. — Il vient de le faire !... Dans le fond, c'est sa façon à lui de s'exprimer...

La fonction graphique

Après cette profonde réflexion métaphysique sur les possibilités de communication entre la matière inerte et la matière vivante, Olivier me déclara tout net :

O. — Et si maintenant on passait au dessin ?

P. — Bien !... Puisque je vois que tu es prêt, je vais t'expliquer ce qui s'appelle « la fonction graphique » !... Comme tu as pu le remarquer sur les touches de ton clavier, tu disposes d'un grand nombre de petits dessins géométriques, de forme et de noirceur différentes.

O. — Oui !... Il y en a des tas !... C'est chouette !

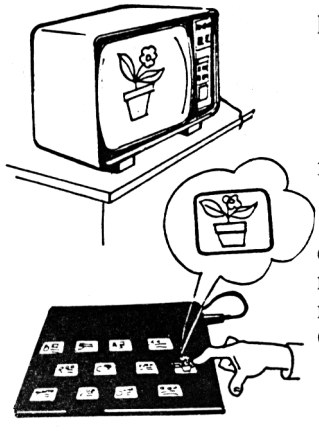
P. — Comme tu dis !... Surtout qu'avec un peu d'imagination, on peut arriver à dessiner à peu près n'importe quoi en les assemblant.

O. — J'imagine qu'il doit encore y avoir une combine atroce pour arriver à les obtenir ?

P. — A peine !... Cela se fait en deux étapes. Il te faut d'abord passer en fonction graphique en appuyant en même temps sur [SHIFT] et sur [GRAPHICS]... Tu sais d'ailleurs que tu es en fonction graphique quand la lettre de ton curseur devient un G.

O. — C'est fait !... Et après ?

P. — Après, il te suffit de maintenir enfoncée la touche [SHIFT] et d'appuyer sur la touche qui comporte le dessin que tu veux reproduire.



O. — Ça y est !... J'ai choisi le damier qui est sur la même touche que la lettre T.

P. — Amuse-toi donc à passer en revue toutes les autres touches, de façon à bien te familiariser avec chaque petit dessin.

Et sur ces bonne paroles je m'éclipse, bien decidé ce coup-ci à m'octroyer au moins cinq grosses minutes de repos... Au bout d'un quart d'heure, l'absence de bruit en provenance de la salle d'ordinateur (le grenier, en l'occurence !) me préoccupa. Cet état anormal, et donc inquiètant, me fit monter l'escalier d'un pas anxieux pour découvrir mon Olivier en train de remplir l'écran de petits dessins.

P. — Alors, on s'amuse bien on dirait ?

O. — Oui, pas mal !... Tu sais que j'ai découvert que si je n'appuyais pas sur la touche [SHIFT], j'obtenais des lettres blanches sur fond noir.

P. — Comme celles du curseur ?

O. — Oui, c'est ça !

P. — Cela s'appelle l'écriture en vidéo inversée. Bravo de l'avoir découverte tout seul !... As-tu noté que même les signes d'opération et de ponctuation peuvent être obtenus de cette façon ?

O. — Oui !... D'ailleurs, je les dessine tous, les uns à la suite des autres (il en était

effectivement a sa cinquième ligne de petits dessins !)... Il n'y a qu'un défaut à tout cela !

P. — Ah, oui ?... Et lequel ?

O. — Plus ça va, et plus il faut de temps à l'ordinateur pour afficher chaque nouveau dessin. Chaque fois que je tape sur une touche, l'écran s'efface, puis il se met à afficher ligne après ligne ce que j'ai déjà écrit. C'est pas génial comme système !

P. — C'est parce que ce n'est pas de cette façon qu'il faut procéder. Tu verras que lorsque tu écriras un programme, il y aura très peu de chances pour qu'une instruction dépasse une ligne entière d'écran. D'ailleurs, pour te montrer que cette façon de procéder n'est pas la bonne, je te mets au défit de trouver un moyen pour effacer l'écran...

O. — Pourtant, cela me paraît simple !... Il me suffit d'appuyer sur [SHIFT] et sur [RUBOUT] !

P. — Et tu n'effaces qu'un caractère à chaque fois, n'est-ce pas ?

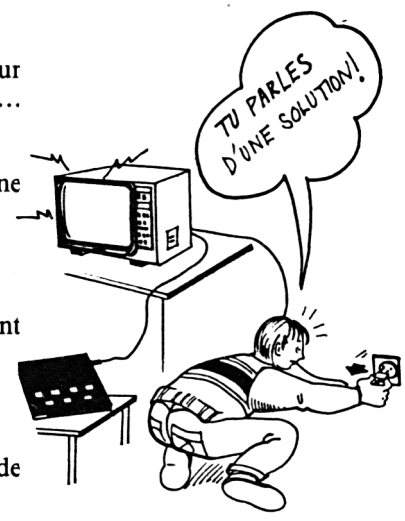
O. — Oui !... Et, en plus, je suis obligé d'attendre que l'ordinateur réinscrive sur l'écran tous les caractères qui restent à effacer avant de pouvoir en effacer un autre !... C'est trop long !

P. — Cela, c'est la preuve que je voulais te fournir !... Heureusement, il existe une solution expéditive.

O. — Ah, oui !... Laquelle ?

P. — Tu débranches la prise de courant de l'alimentation. Cela vide instantanément la mémoire de l'ordinateur.

O. — Tu parles d'une solution !



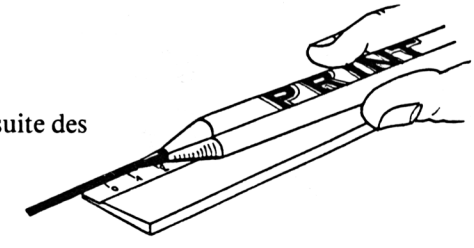
L'opération de nettoyage étant accomplie, Olivier réfléchit un moment avant de m'annoncer :

O. — Et si maintenant, tu me montrais comment on fait pour tracer une ligne droite !

Comment tracer une ligne droite

P. — Ok !... C'est comme si c'était déjà fait !... La première méthode que je te propose est celle que tu as déjà employée tout à l'heure.

O. — ? ? ?



P. — N'as-tu pas dessiné toute une ribambelle de petits carreaux les uns à la suite des autres ?

O. — Si !

P. — Eh bien, si tu répètes toujours le même dessin, tu obtiens une ligne horizontale.

O. — Tiens, c'est vrai ça !... En supposant que je veuille tracer une ligne noire de 4 carreaux de longueur, il me suffit de taper 4 fois le même carré noir ?

P. — Exact !... Au fait, puisque tu en parles, comment fait-on pour obtenir un carré tout noir ?

O. — Ça, je sais !... J'avais remarqué que lorsque j'était en fonction graphique, la touche [SPACE] commandait un carré noir. Dans le fond, c'est pas bête !... C'est l'inverse d'un espace blanc !

P. — C'est bien !... Pour en revenir à ta ligne horizontale, écris donc l'instruction qui réalise cela.

O. — Je suppose qu'il faut que j'utilise à nouveau l'instruction PRINT ?

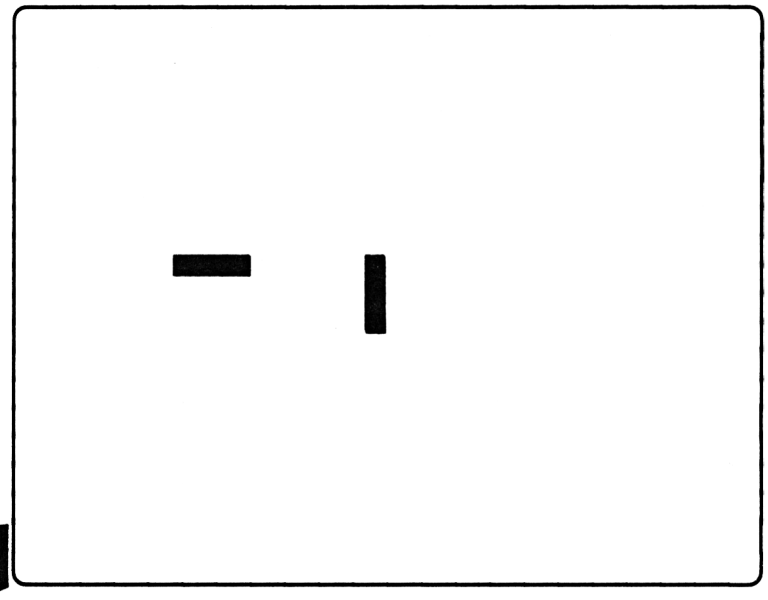
P. — Tu peux même employer l'instruction PRINT AT, si tu souhaites dessiner ta ligne en un endroit précis de l'écran.

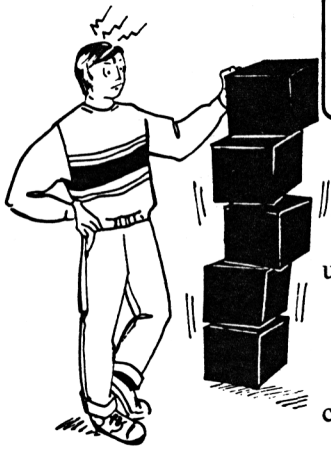
O. — Bon, ça y est !... J'ai fini!... Voilà ce que cela donne :

```
PRINT AT 10,5;"▬▬▬▬"
```

P. — Cela me paraît tout à fait correct !... Tu peux y aller !

Et Olivier appuie sur [NEW LINE] !... C'est alors qu'apparaît sur l'écran, ô sublime récompense de tant d'efforts méritoires, une superbe ligne horizontale.





O. — C'est quand même beau l'informatique...

P. — Attends, tu n'es pas au bout de tes surprises !... Essaye maintenant de dessiner une ligne verticale.

O. — Bonne idée !

Sur ce, je le quitte. Dix minutes plus tard, je questionne sur un ton qui se veut innocent :

P. — Alors, tu y arrives ?

O. — Ça vient, ça vient !

En réalité, je savais que « cela ne pouvait pas venir ». Il lui manquait un élément indispensable que je ne lui avais pas encore fourni. Je lui lançais alors du bas de l'escalier :

P. — Si tu m'expliquais un peu ce qui se passe ?

O. — Eh bien, j'ai du mal à comprendre comment je peux utiliser l'instruction PRINT AT pour tracer une ligne verticale.

P. — Pourtant tu le peux !... En fait, ce qui te gêne, c'est que cette instruction a été prévue au départ pour écrire sur une ligne horizontale.

O. — C'est ça !

P. — Qu'à cela ne tienne !... Puisqu'une telle instruction ne peut écrire que sur une seule ligne, répète donc autant de fois cette instruction que tu as de carreaux verticaux à dessiner.

O. — Mais c'est bien sûr ! (phrase empruntée à une célèbre émission de télévision, qu'il faut ponctuer d'un coup de poing dans la main gauche. Pour les gauchers, c'est l'inverse !)...

L'exécution de cette idée se révéla fort laborieuse car, au bout d'un moment, Olivier me demanda :

O. — Dis donc, Papa, j'ai un problème !... Chaque fois que je tape sur [NEW LINE] pour écrire pour la seconde fois l'instruction PRINT AT, j'ai l'ordinateur qui affiche le résultat de la première et tout est à recommencer.

P. — C'est normal !... Dans un cas comme celui-là, il faut que tu n'utilises qu'une seule instruction PRINT, que tu fais suivre par autant d'instructions AT que tu as de carreaux à dessiner, en n'oubliant pas de les séparer les unes des autres par un point-virgule.

O. — Ok, j'ai compris !... Attends, je recommence tout !... Ça y est, j'ai fini !... Et comme cela, est-ce que ça va ?

```
PRINT AT 10,15;"■";AT 11,15;"■";
AT 12,15;"■";AT 13,15;"■"
```

P. — Affirmatif !... Tu peux maintenant taper sur [NEW LINE] !

O. — Eh bien, c'est bon !... Voilà un petit bout de ligne verticale !... C'est quand même laborieux comme méthode !... J'espère qu'il en existe une autre plus astucieuse, surtout lorsque la ligne à tracer est longue ?

P. — Heureusement, oui !... Tu penses bien que si la ligne fait 20 carreaux de longueur, tu ne vas pas t'amuser (si l'on peut dire !...) à écrire 20 instructions AT les unes à la suite des autres. Seulement, pour pouvoir t'expliquer cette autre méthode, il faut que je te parle avant du « mode programme ».

Le mode programme

O. — Le mode programme ?... Qu'est-ce que c'est ?

P. — C'est la façon d'écrire un programme !... Un programme est une suite d'instructions que l'ordinateur doit exécuter. Jusqu'à présent, tu n'as jamais écrit plus d'une instruction à la fois !

O. — Et pour cause !... Quand j'ai essayé tout à l'heure d'écrire plusieurs fois l'instruction PRINT pour tracer la ligne verticale, l'ordinateur a refusé tout net.

P. — Cela tenait à ce que tes instructions n'étaient pas numérotées. Lorsque tu veux créer un programme, il te faut numéroter chaque ligne d'instruction.

O. — C'est tout ?... Mais c'est très simple !... Et qu'est-ce que je mets comme numéro ?

P. — N'importe lequel entre le 1 et le 9999 !

O. — Tu veux dire qu'il faut que je commence à numéroter ma première ligne d'instructions par le 1, la deuxième par le 2, et ainsi de suite ?

P. — Disons que tu pourrais le faire ainsi, mais que cela ne serait pas très astucieux !

O. — Ah, bon !... Et pourquoi ?

P. — Imagine que tu écrives une série d'instructions en les numérotant ainsi dans l'ordre et que, lorsque tout est fini, tu te rendes compte que tu en as oublié une. Comment vas-tu faire pour l'intercaler entre les autres ?

O. — Ah oui, je comprends ce que tu veux dire !... Il faut que je laisse quelques numéros libres entre chaque instruction, de façon à pouvoir en intercaler d'autres par la suite.

P. — C'est exactement cela !... Je vais même t'indiquer la règle que suivent la plupart des programmeurs ; ils numérotent leurs lignes de dix en dix.

O. — Tant que ça ?... Dans le fond, pourquoi pas ?... Tiens, donne-moi un exemple que j'essaye tout de suite !

P. — Tu n'as qu'à reprendre celui qui ne marchait pas tout à l'heure !

O. — Ah oui, je vois ce dont tu parles !... Il s'agit de celui dans lequel j'ai essayé d'écrire quatre fois de suite l'instruction PRINT !

P. — C'est ça !... Si ce coup-si tu renumérotes chaque ligne, cela devrait marcher !

O. — Ok, j'essaye !... Tiens, mais c'est pas mal !... Chaque fois que j'ai fini d'écrire une ligne et que je tape sur [NEW LINE], elle s'inscrit en haut de l'écran.

P. — C'est ton programme que tu es en train d'élaborer ainsi, et la liste des instructions qui le constituent s'appelle un « listage ».

O. — Ah bon !... Eh bien, ça y est, j'ai fini !... Tout mon programme est entièrement affiché sur l'écran !... Regarde ce que cela donne :

```
10 PRINT AT 10,15;"■"
20 PRINT AT 11,15;"■"
30 PRINT AT 12,15;"■"
40 PRINT AT 13,15;"■"
```

P. — Ok, c'est correct !... Comme maintenant tu es en mode programme, il te faut appuyer sur la touche [RUN], puis ensuite sur la touche [NEW LINE].

O. — Et à quoi correspond la touche [RUN] ?

P. — A une instruction BASIC qui donne l'ordre à l'ordinateur de lancer le programme qu'il a en mémoire. D'ailleurs, « run » en anglais, cela veut dire courir, mais aussi mettre en marche, lancer, etc... Ainsi, quand tu appuies sur cette touche, tu donnes l'ordre à l'ordinateur de lire et d'exécuter le programme que tu viens d'écrire.

Olivier, qui entre temps avait appuyé sur [RUN] et sur [NEW LINE], s'écria :

O. — Ça y est !... Je viens d'obtenir la ligne verticale !... Au fait, cela me fait penser à quelque chose que je voulais te demander depuis un moment : à quoi correspondent les deux chiffres Ø/4Ø qui viennent d'apparaître dans le coin gauche, en bas de l'écran ?

P. — Pour le premier chiffre, tu sais qu'il s'agit d'un code que la machine emploie pour t'indiquer ce qui ne va pas. Dans le cas présent, tout va bien, car le Ø indique que tout le programme s'est déroulé correctement.

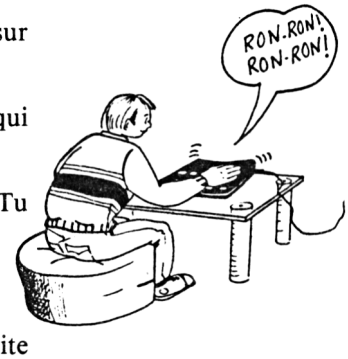
O. — Et le second chiffre ?

P. — Celui-là t'indique le numéro de la dernière ligne exécutée. Ainsi, dans ton cas, tu sais que la ligne 40 est la dernière de ton programme.

O. — Pas mal, pas mal !... Astucieuse cette petite bête-là !...

Et Olivier flatte de la main le boîtier de son ordinateur, comme s'il s'agissait d'un animal domestique...

P. — Maintenant, toi qui es curieux, je te conseille d'appuyer à nouveau sur [NEW LINE] pour voir ce qui va se passer !



O. — Tiens, la ligne noire vient de s'effacer de l'écran et voilà mon programme qui revient à la place. Il ne s'était donc pas effacé ?

P. — Eh, non !... Tu vois, ça, c'est encore un avantage du mode programme ! Tu conserves ton programme en mémoire aussi longtemps que tu le veux.

O. — Et il n'y a pas moyen de l'effacer ?

P. — Si, bien sûr !... Il te suffit pour cela d'appuyer sur la touche [NEW] et ensuite sur [NEW LINE].

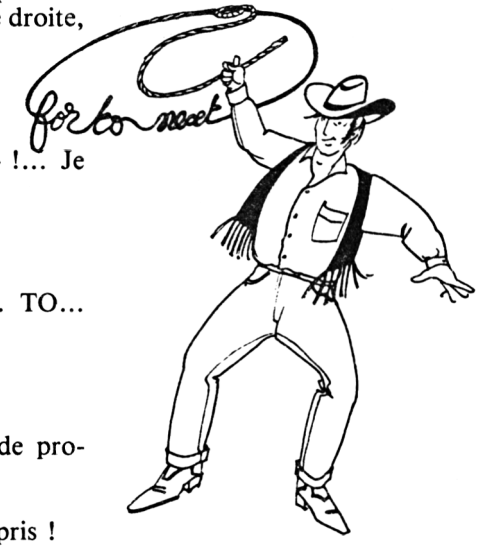
O. — Et crac !... Ça y est, j'ai tout effacé !... Dis donc, il ne s'agit pas d'appuyer par erreur sur la touche [NEW] ?

P. — Comme tu dis !... Il vaut mieux réfléchir à deux fois avant d'effacer ce qui peut représenter des heures de travail !

O. — Bon !... Je ne nie pas l'intérêt de tout ceci, mais si tu me montrais maintenant comment on fait pour tracer « astucieusement » une ligne horizontale, surtout lorsque sa longueur dépasse plusieurs carreaux.

Les horizontales

P. — Maintenant que tu sais comment fabriquer un programme, je vais pouvoir t'expliquer comment on peut simplifier le travail d'écriture du traçage d'une ligne droite, même si elle fait toute la largeur de l'écran.



O. — Vas-y, je t'écoute...

P. — Il s'agit de créer ce qui, en informatique, s'appelle une « boucle » !... Je t'explique : une boucle est une série d'instructions qui se mordent la queue.

O. — C'est très clair !... Tu peux préciser ?

P. — Bien sûr !... Je vais prendre comme exemple l'instruction FOR... TO... NEXT.

O. — Qu'est-ce que c'est que ça ?

P. — C'est une instruction BASIC qui permet de parcourir une boucle de programme autant de fois que tu lui dis de le faire.

O. — Tu vas peut-être dire que je suis bouché, mais j'ai toujours rien compris !

P. — Supposons que tu veuilles tracer une ligne horizontale sur toute la largeur de l'écran !

O. — Ça tombe bien, c'est justement ce que je voudrais faire !

P. — Or, si tu te souviens bien, l'écran comporte 32 colonnes dans le sens de la largeur...

O. — ... et 22 lignes dans le sens de la hauteur, je sais !

P. — Et bien, plutôt que de taper le dessin du carré noir autant de fois qu'il y a de colonnes, c'est-à-dire 32, il te suffit de l'écrire une seule fois, en précisant à la machine que tu souhaites qu'elle le dessine 32 fois, une par colonne.

O. — Je commence à entrevoir le quart du tiers d'une faible lueur...

P. — Pour arriver à cela, tu vas donc devoir définir une variable qui devra obligatoirement être une lettre unique.

O. — Comme un X par exemple ?

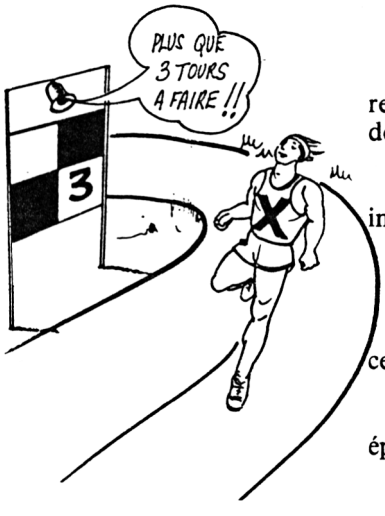
P. — C'est ça !... Tu indiqueras ensuite à l'aide de l'instruction FOR... TO à partir de quelle colonne tu débutes ta ligne et à laquelle tu la termines.

O. — Attends, j'ai compris !... Tu me dis si je me trompe !... Si je veux dessiner une ligne horizontale qui aille de la colonne Ø à la colonne 31, je n'ai qu'à écrire quelque chose dans le genre :

```
10 FOR X=Ø TO 31
```

P. — Pas mal, pas mal !... Si maintenant tu rajoutes l'instruction NEXT X à la fin de ta boucle, puisque tu as choisi X comme variable, tu obliges ainsi la variable X à changer de valeur à chaque tour.

O. — Et je suppose que je place, entre ces deux lignes d'instruction, ce que je veux dessiner ?



P. — Exact !...

O. — Donc j'écris quelque chose comme PRINT AT, suivi du motif que je veux reproduire !... Mais au fait, j'ai un problème !... Comment est-ce que j'inscris les coordonnées de traçage, puisque le numéro de la colonne change tout le temps ?

P. — Tu le remplaces par sa variable !... C'est pour cela que les variables ont été inventées !... Au fait, sur quelle ligne veux-tu dessiner ton horizontale ?

O. — Tout en haut de l'écran, c'est-à-dire sur la ligne Ø.

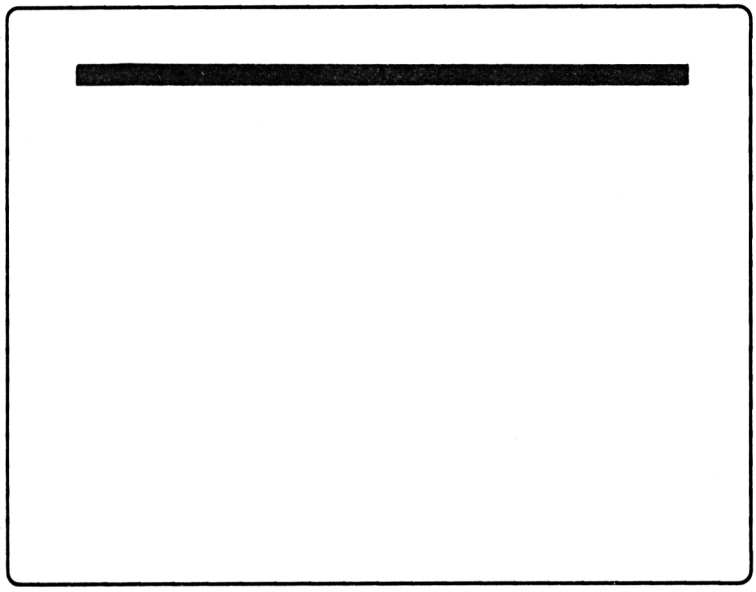
P. — Bien !... Maintenant, je te laisse finir l'écriture de ta boucle !... Rappelle-moi ce que tu utilises comme motif ?

O. — Toujours le carré noir, de façon à dessiner une belle horizontale, bien épaisse !... Ça y est, j'ai fini !... Voilà ce que cela donne :

```
20 PRINT AT Ø,X;"■"
30 NEXT X
```

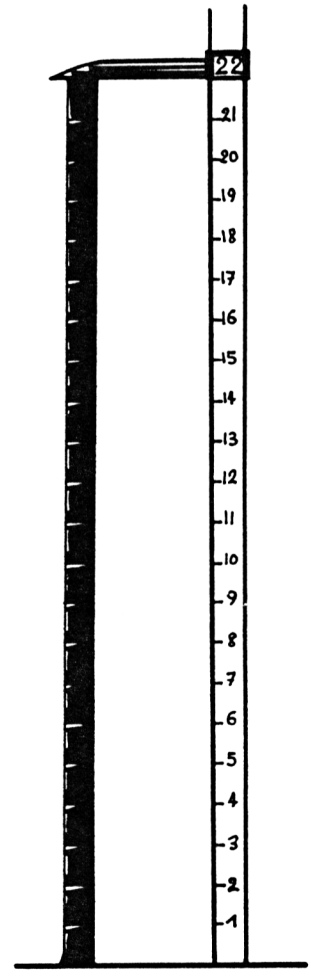
P. — Compliments, c'est exact du premier coup !... Je te laisse lancer toi-même ton programme. Tu vas voir comme c'est « chouette » de regarder l'ordinateur dessiner tout seul.

O. — Attention, c'est parti !... J'appuie sur [RUN] et ensuite sur [NEW LINE] !... Whaooo !... C'est super !



Et effectivement, cela pouvait être « super ». Le maître venait de commander à son esclave de dessiner pour lui et l'esclave obéissait au doigt et à l'œil. Quelle puissance que celle de l'ordinateur ! Olivier ne cessait de taper sur [RUN] et sur [NEW LINE] pour voir la ligne se dérouler d'un bord à l'autre de l'écran. Ce petit jeu dura quelques minutes puis, subitement lassé, il se tourna vers moi et me déclara :

O. — Et si maintenant on passait aux verticales !



Les verticales

P. — Cela me paraît être la suite logique des événements !... A ton avis, par quoi « on » commence ?

O. — Je ne sais pas trop !... A première vue, il me semble qu'il ne doit pas y avoir beaucoup de différences entre la façon de tracer des horizontales ou des verticales.

P. — Effectivement, ton raisonnement est bon !... Il y a même si peu de différence, que nous allons les examiner ensemble tout de suite !

O. — Je suppose que la première concerne la hauteur maximale qu'une verticale peut avoir ...?

P. — Vas-y, continue !

O. — Comme l'écran ne dispose que de 22 lignes, je suppose que je dois modifier la première instruction en conséquence !... Que penserais-tu de :

```
10 FOR X=0 TO 21
```

P. — Très bien !... Ta variable X représente dans ce cas le nombre de lignes sur lesquelles tu vas répéter le même motif.

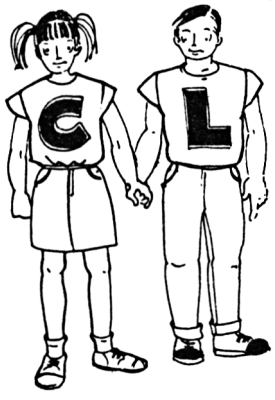
O. — Dis donc, cela ne te gêne pas d'employer la même lettre X pour désigner à la fois des lignes ou des colonnes ?

P. — Non !... Mais je comprends que cela puisse te dérouter !... Pourquoi dans ce cas ne pas choisir la lettre Y ?

O. — Tu ne trouves pas que tous ces X et ces Y, cela ressemble un peu trop à ce que je fais à l'école ?

P. — Qu'à cela ne tienne !... Je te propose alors la lettre C pour la variable des colonnes et la lettre L pour celle des lignes.

O. — Bonne idée !... Et comme ça, cela sera plus facile à retenir !... Bien, maintenant tu ne me dis plus rien !... Je tiens à y arriver tout seul !



Je laisse donc Olivier écrire tout seul son programme, écriture qui se révéla ponctuée de commentaires désobligeants envers sa personne en particulier et l'informatique en général. Finalement il termina et, me montrant son œuvre, il me demanda :

O. — Si tu veux bien regarder ce j'ai écrit !... Je crois que ce doit être correct !

```
10 FOR L=0 TO 21
20 PRINT AT L,0;"■"
30 NEXT L
```

P. — Effectivement, cela me paraît bien écrit !... Je vois que tu as choisi la variable L pour désigner les lignes et que tu souhaites dessiner ta verticale tout à fait à gauche de l'écran dans la colonne 0.

O. — Alors, si tout est bon, j'envoie !

Et comme précédemment, l'ordinateur dessina de lui-même une superbe ligne verticale. On a beau y être habitué, c'est toujours fascinant de voir une machine faire cela !... Olivier me ramena à la réalité :

O. — Et si on dessinait autre chose ?

P. — Tiens, pourquoi pas ?... Ce ne sont pas les motifs qui nous manquent !... Qui ne nous deux commence ?

O. — Moi !... Je vais t'en dessiner une plus étroite en me servant du motif de la touche [5].

P. — Ça, c'est une bonne idée !... Et comment vas-tu procéder ?

O. — Ben, cette question !... Je vais tout réécrire !

P. — Tu vas tout réécrire alors qu'il te suffit de remplacer un caractère par un autre ?

O. — Tiens, c'est vrai !... Je n'y avais pas pensé !... Tu me montres comment on fait pour effectuer une correction ?

P. — Il faut pour cela que tu « édites » la ligne que tu veux corriger, c'est-à-dire qu'il faut que tu la sortes du programme. Pour cela, tu dois d'abord amener le curseur en face de la ligne à éditer.

O. — Je suppose que ce que tu appelles le curseur, c'est ce « machin » noir avec un signe « supérieur à » dedans ?

P. — C'est cela !

O. — Pour le moment, il est en face de la ligne 3Ø.

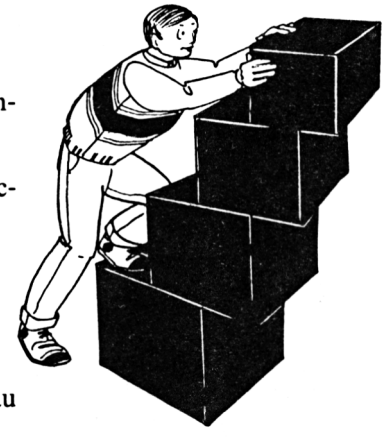
P. — C'est normal, car il se place automatiquement en face de la dernière ligne que tu as rentrée !... Pour le faire monter, il te suffit d'appuyer en même temps sur la touche [SHIFT] et sur celle qui porte la flèche ⇧ dirigée vers le haut.

O. — Ok, ça marche !... Et pour le faire descendre, j'imagine que c'est la touche d'à côté marquée de la flèche ⇩ ?

P. — Tout à fait cela !... Bien, quand tu auras fini de jouer à l'ascenseur, tu placeras ton curseur juste en face de la ligne 2Ø, celle que tu veux éditer.

O. — Ça y est, c'est fait !

P. — Appuie maintenant sur les touches [SHIFT] et [EDIT] !... Que se passe-t-il ?



O. — La ligne 2Ø vient d'apparaître en bas de l'écran.

P. — Elle vient d'être éditée !... Tu peux maintenant la modifier comme bon te semble.

O. — Ça y est, j'ai terminé !... Regarde !... J'ai remplacé le carré noir par un rectangle.

```
20 PRINT AT L,0;"▮"
```

P. — C'est très bien tout cela !... Il ne te reste plus qu'à rentrer cette ligne à nouveau dans le programme, à la place de l'ancienne ligne 2Ø.

O. — Et qu'est-ce que je dois faire pour cela ?

P. — Rien d'autre que ce que tu fais après avoir terminé d'écrire une ligne !... Tu appuie sur [NEW LINE] !

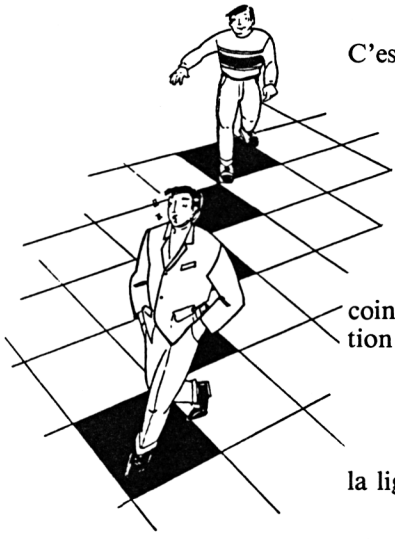
O. — Ah bon !

Et c'est sur cette dernière réflexion qu'Olivier plongea la tête la première dans l'élaboration de verticales aux motifs les plus variés. Tout y passa : le rectangle vertical, le carré hachuré, le petit carré, le signe plus, l'étoile, les deux points, et que sais-je encore ?...

Les obliques

Toutes les bonnes choses ayant une fin, celle-ci se précisa lorsque Olivier, ayant épuisé tous les motifs que l'ordinateur mettait à sa disposition, vint me trouver et me dit :

O. — Et pour les lignes obliques, comment fait-on ?



P. — On va commencer par dessiner une ligne oblique descendante à 45 degrés !... C'est la plus facile !... Après, on compliquera un peu !

O. — Ok !

P. — Tu vois à peu près comment il faut procéder ?

O. — Non, pas du tout !

P. — Bon !... Alors, je suppose que tu commences à dessiner ton oblique à partir du coin supérieur gauche de l'écran. Tu vas donc débuter par le carreau qui est à l'intersection de la colonne Ø et de la ligne Ø.

O. — Je te suis...

P. — Le carreau suivant sera celui qui est placé à l'intersection de la colonne 1 et de la ligne 1, puis à l'intersection de la colonne 2 et de la ligne 2, et ainsi de suite...

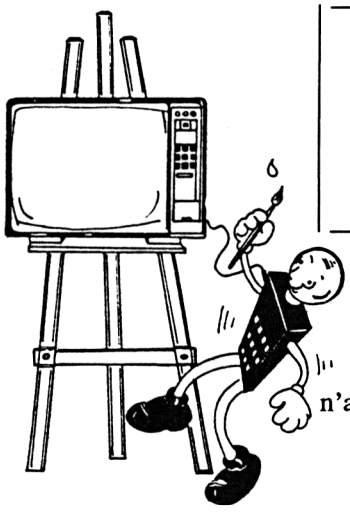
O. — Ça y est, j'ai compris !... Cette fois, au lieu d'introduire une seule variable dans l'instruction PRINT, il en faut deux !

P. — Et voilà !... Tu vois, ce n'est guère plus compliqué à programmer qu'une simple ligne verticale ou horizontale !

O. — Oui, mais attends !... Si je dois introduire deux variables, il faut que j'écrive également deux fois l'instruction FOR... TO !

P. — Tu le pourrais, mais cela ne serait pas astucieux !... Si tu regardes bien, tu vas te rendre compte que tu n'as pas réellement besoin de deux variables, mais en réalité de deux fois la même. Quand tu progresses d'un carreau par colonne, tu progresses également d'un carreau par ligne.

O. — Oh, mais c'est encore plus simple que je croyais !... Laisse-moi deux minutes pour l'écrire !... Voilà, c'est fait !... Qu'est-ce que tu en penses ?



```
10 FOR X=Ø TO 21
20 PRINT AT X,X;"■"
30 NEXT X
```

P. — C'est parfait !... Au fait, pourquoi avoir choisi la lettre X ce coup-ci ?

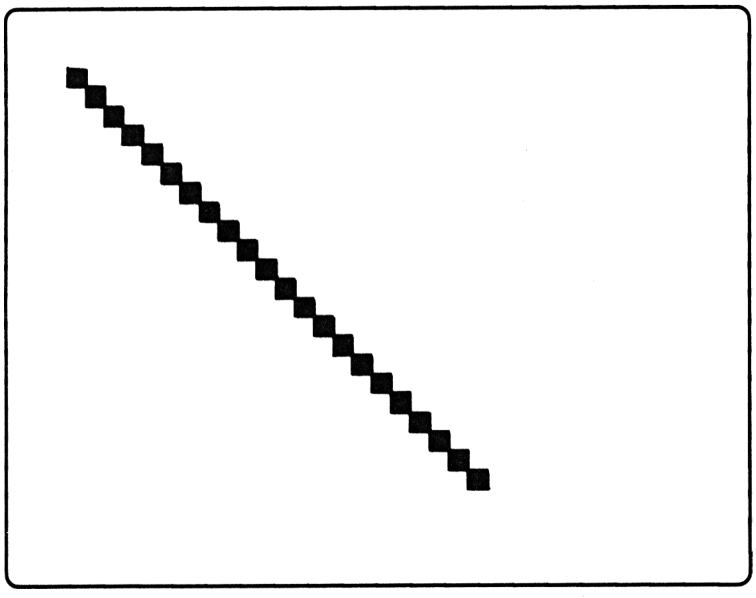
O. — Ben, comme la variable devait représenter à la fois une ligne et une colonne, je n'ai pas su choisir entre L ou C !... Alors, j'ai pris X, car c'était plus simple !

P. — Une autre question : pourquoi t'être limité à 21 ?

O. — Parce que je ne dispose que de 21 lignes !... Comme les lignes et les colonnes défilent en même temps, j'ai dû me contenter du plus petit nombre.

P. — Bravo !... Tu appuies un petit coup sur [RUN] et [NEW LINE] pour voir ce que cela donne !

Et à nouveau le miracle s'accomplit ! L'ordinateur traça tout seul une ligne oblique qui, partant du coin supérieur gauche de l'écran, le traversa en diagonale. Je lui conseillais :



P. — Si tu veux tracer une oblique plus mince, tu n'as qu'à employer le damier de la touche [Y]

O. — Tiens, c'est pas mal comme idée !... Il y a deux fois plus de petits carrés, ce qui fait apparaître le trait plus mince !... Merci, je note !

P. — Bien !... Et maintenant, si on passait à la suite, c'est-à-dire si on dessinait une ligne oblique montante à 45 degrés ?

O. — De quel carreau veux-tu que je démarre ?

P. — Pars du coin inférieur gauche de l'écran !

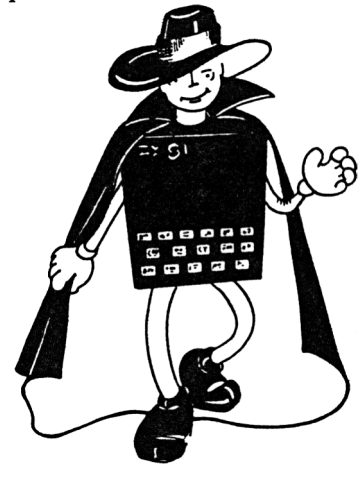
O. — Bon, alors attends que je traduise... Cela correspond à la colonne Ø et à la ligne 21.

P. — C'est cela, et le carreau suivant correspond à la colonne 1 et à la ligne 2Ø. Comme tu peux le remarquer cette fois, au fur et à mesure que le numéro des colonnes augmente, celui des lignes diminue de la même quantité.

O. — Tout cela ne me paraît pas insurmontable !... Il suffit que je soustraie la variable X du numéro de la ligne de départ pour obtenir une coordonnée de ligne qui diminue quand celle des colonnes augmente.

P. — Ce qui donne ?

O. — Attends, j'écris !... Voilà, tu peux regarder !

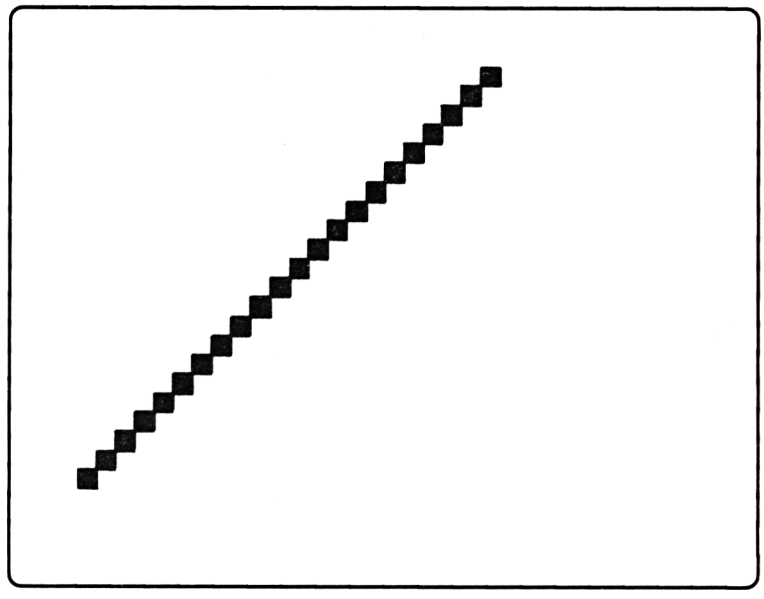


```
10 FOR X=Ø TO 21
20 PRINT AT 21-X,X;"■"
30 NEXT X
```

P. — Chapeau !... Et à l'écran, qu'est-ce que cela donne ?

O. — Un instant, j'appuie sur [RUN] et sur [NEW LINE] !

Et une ligne oblique, qui ressemblait comme un reflet à sa sœur jumelle, se déroula à son tour sur l'éran.



P. — Comme précédemment, tu peux changer de motif et prendre le damier de la touche [T], si tu veux obtenir une ligne moins épaisse.

O. — C'est comme si c'était fait !

Et Olivier repassa en revue les divers motifs en sa possession afin de déterminer de lui-même ce qui convenait le mieux à une oblique. Soudain, alors que je m'y attendais le moins, il s'exclama :

O. — Je viens d'avoir une idée !

P. — Aïe ! (plaisanterie qu'il ne goûte guère !).

O. — J'imagine qu'on doit pouvoir enchaîner le tracé d'une oblique descendante avec celui d'une montante, n'est-ce pas ?

P. — Gagné !... Il te suffit pour cela d'écrire les deux programmes l'un à la suite de l'autre.

O. — Attends, j'entrevois un problème !... Comme je ne dispose que de 32 colonnes pour les deux obliques, je vais devoir limiter le nombre de lignes à 16.

P. — Tout à fait exact !... Décidément, je ne sers plus à grand chose !

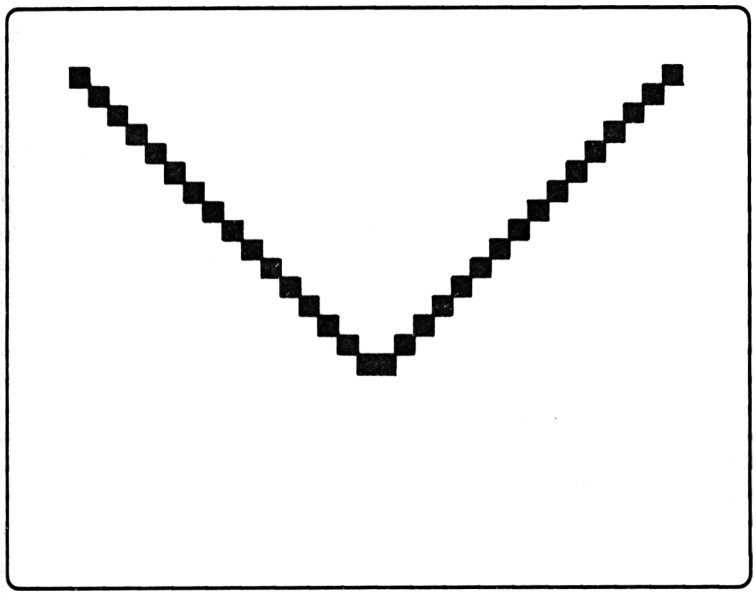
O. — Mais si, mais si !

Tout en me consolant, Olivier tapait déjà sur son clavier. Au bout d'un moment, il me demanda :

O. — Dis-moi ce que tu penses de cela ?

```
10 FOR X=0 TO 15
20 PRINT AT X,X;"■"
30 NEXT X
40 FOR X=16 TO 31
50 PRINT AT 31-X,X;"■"
60 NEXT X
```

P. — J'en pense que cela ne me paraît pas génial outre mesure, mais j'imagine que cela doit pouvoir marcher !



Et effectivement, cela « marcha » ! L'ordinateur dessina un immense V majuscule en travers de l'écran. Fort de ce résultat prometteur, Olivier m'annonça qu'il allait essayer maintenant de dessiner tout seul des obliques à 30 et à 60 degrés, ce dont je lui sus gré, l'après-midi touchant à sa fin et le gazon étant toujours à tondre...

Les pavés

Plus tard dans la soirée, au moment du repas, je questionnais Olivier :

P. — Alors, et ces obliques à 30 et 60 degrés ?

O. — Ça penche, ça penche !... Tu sais, c'était tout simple !... Il suffisait que je multiplie ou que je divise par deux l'une des variables par rapport à l'autre et le tour était joué !... Vraiment pas de quoi fouetter un chat !

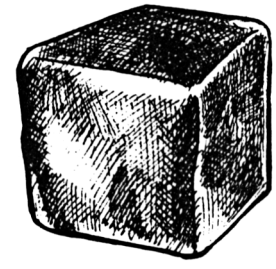
P. — Et maintenant, que veux-tu faire ?

O. — Depuis un moment, je me torture les méninges pour tenter de trouver une solution pour dessiner un carré plein !

P. — Un grand carré ?

O. — Oui, de 10 × 10 carreaux par exemple !

P. — Et qu'est-ce qui t'arrête ?



O. — Ce qui m'arrête, c'est que la solution que j'ai trouvée ne me paraît pas géniale !

P. — Explique !

O. — Je me suis dit que pour dessiner un carré de 10 × 10, je pouvais dessiner 10 lignes horizontales de 10 carreaux chacune, les unes en dessous des autres...

P. — ... ou bien 10 colonnes verticales de 10 carreaux chacune, les unes à côté des autres !

O. — C'est ça !... Seulement, écrire dix fois de suite les instructions FOR... TO, PRINT AT et NEXT ne me semble pas être une solution simple.

P. — Et elle ne l'est pas !... La solution est dans ce qui s'appelle les « boucles emboîtées » !

O. — Les boucles emboîtées ?... Qu'est-ce que c'est que cette nouveauté ?

P. — Ce n'est pas à proprement parler une nouveauté !... Tout au plus, c'est une extension de ce que tu sais déjà sur les boucles !

O. — Vas-y, je t'écoute !

P. — Il te faut donc dessiner 10 lignes de 10 carreaux chacune. Si chaque ligne ne comportait qu'un seul carreau au lieu des dix, tu te retrouverais à écrire un programme de dessin d'une ligne verticale ! Exact ?

O. — Exact, mais je ne vois pas où tu veux en venir !

P. — Ecris donc le programme, cela va t'aider à comprendre !

O. — C'est comme si c'était fait !... Le temps de taper les trois instructions !... Et voilà le programme de « Monsieur » !

```
10 FOR L=1 TO 10
20 PRINT AT L,0;"■"
30 NEXT L
```

P. — Tu vois donc qu'entre la ligne d'instruction numéro 10 et la ligne 30, il existe une instruction capable de dessiner une ligne de 1 carreau. Il te suffit de la remplacer par un petit programme capable de dessiner une ligne de 10 carreaux. Tu n'en connaîtrais pas un, par hasard ?

O. — Si, bien sûr !... Celui que j'ai déjà employé pour les lignes horizontales.

P. — C'est ça !... Il te reste donc à placer entre la ligne 10 et la ligne 30 le programme de dessin d'une ligne horizontale de 10 carreaux.

O. — Ok, j'ai compris !... Pour les numéros des lignes d'instructions, je peux mettre ce que je veux ?

P. — Bien sûr !

O. — Bon, bon !...

Un moment plus tard, Olivier se tourne vers moi et, me montrant l'écran, il me demande fièrement :

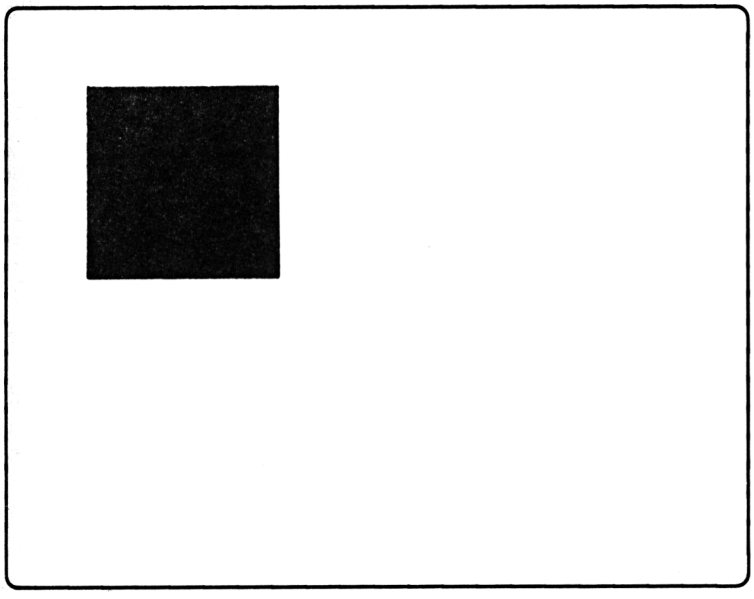
O. — Alors, que penses-tu de cela ?

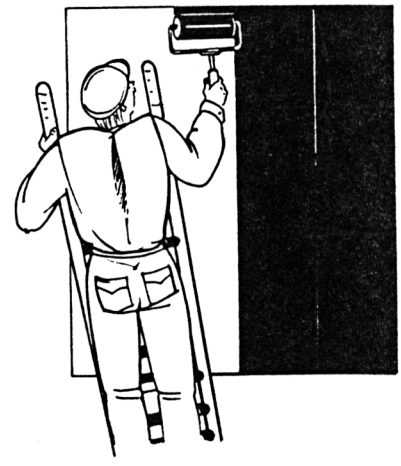
```
10 FOR L=1 TO 10
15 FOR C=1 TO 10
20 PRINT AT L,C;"■"
25 NEXT C
30 NEXT L
```

P. — J'en pense que c'est parfait !

O. — Ouf !... Je n'étais pas très sûr au sujet des coordonnées de l'instruction PRINT. Mais je me suis dit que, comme il y avait plusieurs lignes et plusieurs colonnes, il me fallait utiliser à la fois la variable L et la variable C.

P. — Et tu as très bien fait !... Amuse-toi donc à regarder ce que cela donne sur l'écran !





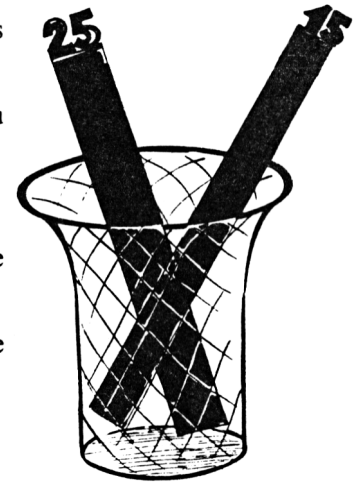
O. — C'est chouette, n'est-ce pas ?... J'imagine maintenant qu'il me suffit d'inverser la ligne 1Ø avec la ligne 15 pour obtenir la même chose, mais dessinée colonne par colonne ?

P. — Oui, bien sûr, mais à condition d'inverser également les lignes 25 et 3Ø. Comme tu as pu le remarquer, tu as deux boucles de programme, l'une étant contenue dans l'autre. Or, il faut qu'il en soit toujours ainsi !

O. — Ah bon, une boucle doit jamais en couper une autre ?

P. — Non !... Ça, c'est une règle absolue. Tu peux placer autant de boucles que tu veux dans un programme, mais il faut qu'elles soient toutes « emboîtées » les unes dans les autres.

O. — Ah, je comprends maintenant pourquoi tu me parlais au début de boucles emboîtées !... Mais au fait, comment fait-on pour inverser les lignes 1Ø et 15 ?



P. — Tu peux carrément les renuméroter !... Mais si tu es aussi fainéant que moi, tu vas t'apercevoir qu'il suffit de n'en renuméroter qu'une seule.

O. — Ah bon !... Et laquelle ?

P. — La 15 par exemple !... Si tu la transformes en ligne 5, elle passera avant la ligne 1Ø et, ainsi, tu n'auras pas à toucher à la ligne 1Ø.

O. — C'est astucieux ça, dis donc !... Je vais pouvoir faire pareil avec la ligne 25 que je vais transformer en ligne 35 de façon à ne pas toucher à la ligne 3Ø.

P. — Bravo, je vois que tu as compris !

O. — Il me reste toutefois un problème à résoudre !

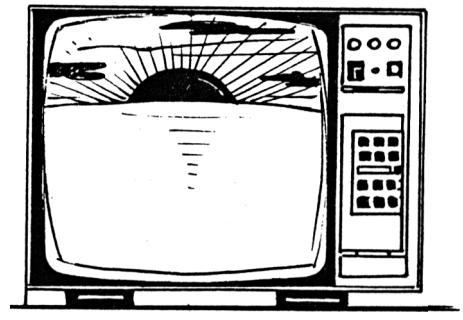
P. — Lequel ?

O. — Maintenant que j'ai écrit les nouvelles lignes 5 et 35, comment puis-je me débarrasser des anciennes lignes 15 et 25 ?

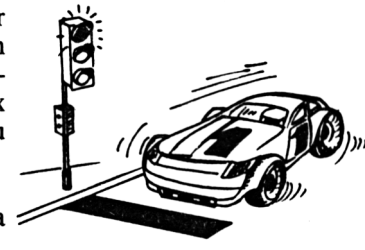
P. — Très bonne question !... Il suffit pour effacer une ligne, comme la ligne 15 par exemple, de taper le numéro 15 et d'appuyer sur [NEW LINE].

O. — C'est aussi simple que cela ?

Et c'est sur cette dernière réplique que se termina cette première journée d'informatique dans laquelle Olivier découvrit quelques-unes des subtilités du dessin sur l'écran d'un micro-ordinateur.



LES AIRES DE JEUX



Pendant toute la semaine qui suivit cette première journée mémorable, Olivier s'entraîna, dès qu'il avait un instant libre, à refaire de lui-même les programmes de dessin qu'il avait appris. Manifestement, il les avait totalement assimilés (et passablement modifiés !) et n'attendait plus qu'un feu vert de ma part pour se lancer à l'assaut de nouveaux horizons. J'en eus d'ailleurs la confirmation lorsque, le vendredi soir au moment du repas, il me demanda sur un ton qui ne trompait guère sur son impatience :

O. — Tu auras le temps de consacrer quelques minutes demain pour m'enseigner la suite ?

Quelques minutes !... Doux euphémisme pour qualifier ce qui durerait facilement plusieurs heures !...

P. — La suite de quoi, lui répondis-je sur un ton innocent.

O. — Ben voyons !... La suite de la façon de programmer ses jeux vidéo !... Je voudrais maintenant apprendre à dessiner autre chose que des lignes droites !

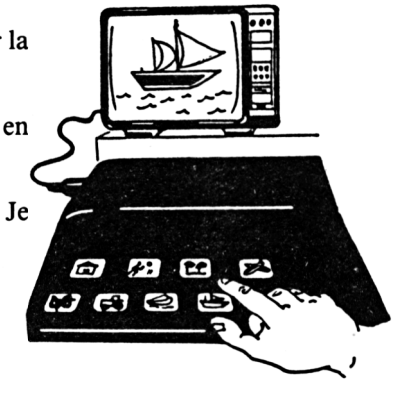
P. — Quoi, par exemple ?

O. — Est-ce que je sais, moi ?... Des véhicules spatiaux, des avions, des bateaux, des voitures de course, des décors, des personnages ; bref, tout ce qui peut constituer un jeu vidéo !

P. — Quel programme, dis donc !... Et tu voudrais qu'on commence quand ?

O. — Pourquoi pas dès demain matin ?... Tu es libre n'est-ce pas ?

Je l'étais, ou plutôt je ne l'étais plus. Il ne fallait pas que je me fasse d'illusions sur la journée du lendemain. Elle y passerait tout entière !



Nous démarrâmes donc le lendemain matin de bonne heure. Olivier qui paraissait en grande forme me proposa :

O. — Montre-moi comment on fait pour dessiner un cadre tout autour de l'écran. Je voudrais m'en servir pour en faire un terrain de jeux !

Le pourtour d'un terrain

P. — Il me semble que si tu te remémores ce que nous avons déjà vu ensemble la semaine dernière, tu devrais être capable de dessiner tout seul un cadre autour de l'écran.

O. — ? ? ?

P. — Il te suffit de dessiner deux lignes horizontales, l'une en haut et l'autre en bas de l'écran, suivies par deux lignes verticales à gauche et à droite !

O. — Ah oui, c'est vrai !... Il suffit que j'écrive quatre fois consécutives les instructions FOR...TO, PRINT et NEXT pour cela, une fois par ligne à tracer !

P. — Moi, je n'en serais pas si sûr que cela !... Cela ne te paraît pas un peu trop lourd comme méthode ?

O. — En tout cas, c'est la seule que je connaisse !

P. — Alors, je te propose de dessiner avec les mêmes instructions FOR...TO et NEXT les deux lignes horizontales d'un seul coup, en plaçant entre elles deux instructions PRINT.

O. — Comme cela ?

```
10 FOR C=0 TO 31
20 PRINT AT 0,C;"■"
30 PRINT AT 21,C;"■"
40 NEXT C
```

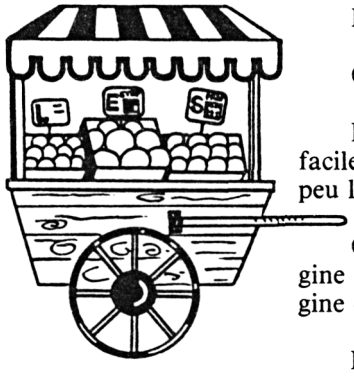
P. — Oui !... Tu peux même employer la variante qui consiste à n'employer qu'une seule instruction PRINT pour dessiner les deux lignes, en séparant chaque série de coordonnées par un point virgule !

O. — Ah oui, je me souviens !... Je l'ai fait quand je ne connaissais pas le mode programme !... Et on peut l'utiliser ici ?

P. — Tout à fait !

O. — Est-ce que cela ressemble à quelque chose comme ça ?

```
20 PRINT AT 0,C;"■";AT 21,C;"■
"
```



P. — Oui, tout à fait !...

O. — Laquelle des deux est préférable ?

P. — A quelque chose près, elles se valent !... La première présentation est plus facile à modifier car les lignes sont plus courtes. La seconde, par contre, économise un peu l'espace nécessaire en mémoire.

O. — Alors, je préfère la seconde !... Comme le micro-ordinateur ne dispose à l'origine que de 1 Kmot de mémoire, c'est-à-dire de 1000 emplacements de mémoire, j'imagine qu'il vaut mieux toujours choisir la solution la plus économique !

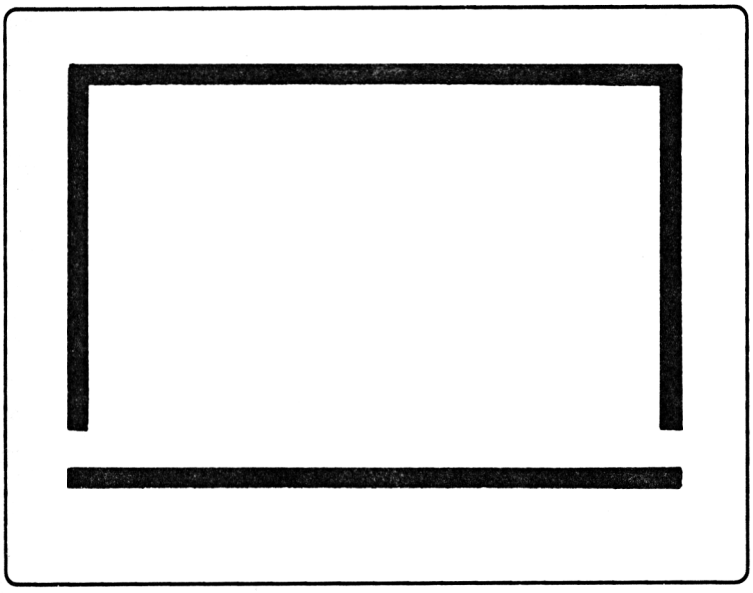
P. — Cela me paraît être un choix judicieux !

O. — Maintenant, pour dessiner les verticales, je sais comment il faut procéder !... J'écris :

```
50 FOR L=0 TO 21
60 PRINT AT L,0;"■";AT L,31;"■
"
70 NEXT L
```

P. — C'est parfait !... Il ne te reste plus qu'à vérifier ce que cela donne sur l'écran !

Et un superbe cadre se dessina autour de l'écran, en commençant par les deux lignes horizontales, suivies aussitôt par les deux lignes verticales, sauf que, au bas de chaque verticale, il manquait deux carreaux !



O. — Ben, qu'est-ce qui lui arrive à l'ordinateur ?... Il est fatigué de dessiner ?

P. — C'est à peu près ce qui se passe !... Quel est le message d'erreur ?

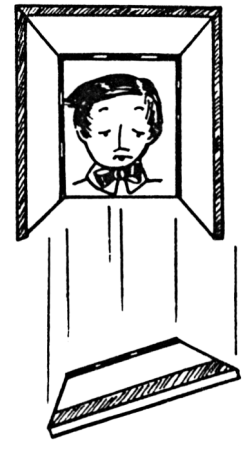
O. — Attends, je regarde !... 4/7Ø !

P. — Le chiffre 4 indique que la mémoire est saturée. En fait, il ne s'agit dans ce cas que de la partie de la mémoire affectée à l'affichage de l'écran.

O. — Et qu'est-ce qu'on peut faire pour y remédier ?

P. — Peu de choses !... Ou bien acheter une mémoire de 16 Kmots qui peut se rajouter dessus et multiplie sa capacité de mémoire par seize, ou bien réduire un peu les dimensions de ton cadre !

O. — Tu m'achètes la mémoire 16 K ?



P. — Il n'en est pas question dans l'immédiat !... Il vaut mieux que tu apprennes d'abord à te servir à fond de celle de 1 K que tu possèdes, avant de penser à en changer !

O. — Bon, j'ai compris !... Il faut donc que je réduise la taille de mon cadre !

P. — Eh oui, j'en ai bien peur !... Ou alors, que tu découvres une astuce pour écrire un programme de dessin de ton cadre qui emploie moins d'instructions !

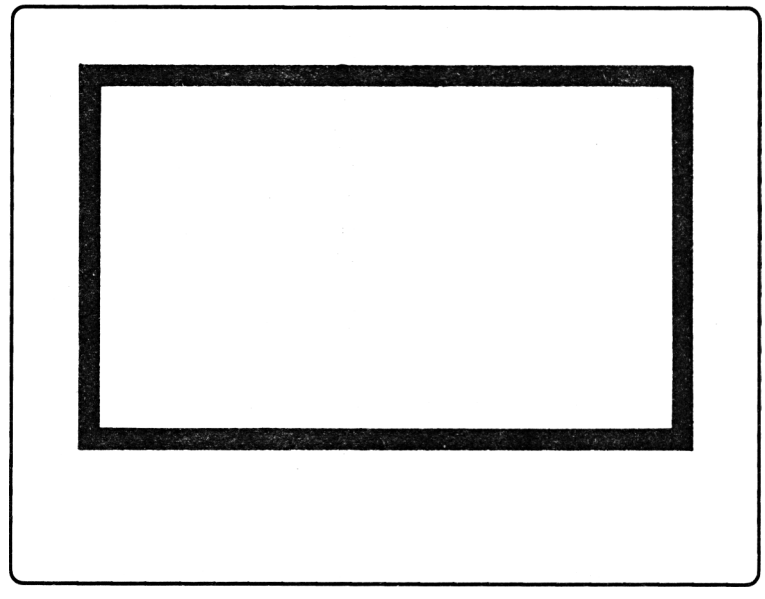
O. — Passons !... Dis-moi, comme il me manque deux carreaux au bas de chaque verticale, il suffit que je remonte de deux carreaux la ligne du bas !

P. — Eh oui !... Ou que tu baisses celle du haut de deux carreaux également !

O. — Attends, pas si vite !... Voilà déjà ce qui devrait permettre de fermer le cadre :

```
10 FOR C=Ø TO 31
20 PRINT AT Ø,C;"■";AT 19,C;"■
"
30 NEXT C
40 FOR L=Ø TO 19
50 PRINT AT L,Ø;"■";AT L,31;"■
"
60 NEXT L
```

O. — Je n'ai eu qu'à modifier les lignes d'instruction numéro 2Ø et 4Ø !... Tu noteras que j'en ai profité pour renuméroter dans l'ordre toutes les autres lignes !



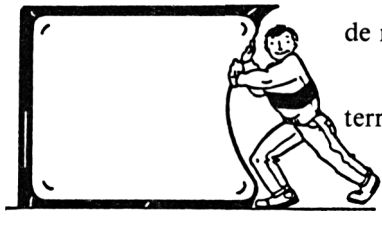
P. — Très bien !... Maintenant, au tour de celle du haut !

O. — Il faut à nouveau que je modifie les lignes 2Ø et 4Ø !... Ça y est, c'est fini !

```
20 PRINT AT 2,C;"■";AT 21,C;"■
"
40 FOR L=2 TO 21
```

P. — Qu'est-ce que tu obtiens ?

O. — La même chose !... Je peux donc déplacer comme je veux un dessin vers le haut ou le bas de l'écran !



P. — Exact !... Maintenant, je voudrais te faire remarquer que tu n'as pas envisagé de réduire ton cadre dans le sens de la largeur de l'écran !

O. — Tiens, c'est vrai !... Il suffit que je supprime deux colonnes et j'obtiendrais un terrain encore potable !

P. — Et tu les supprimes où tes deux colonnes ?

O. — Ne me dis pas qu'il y a une différence entre le côté droit et le côté gauche de l'écran ?

P. — Eh si !...

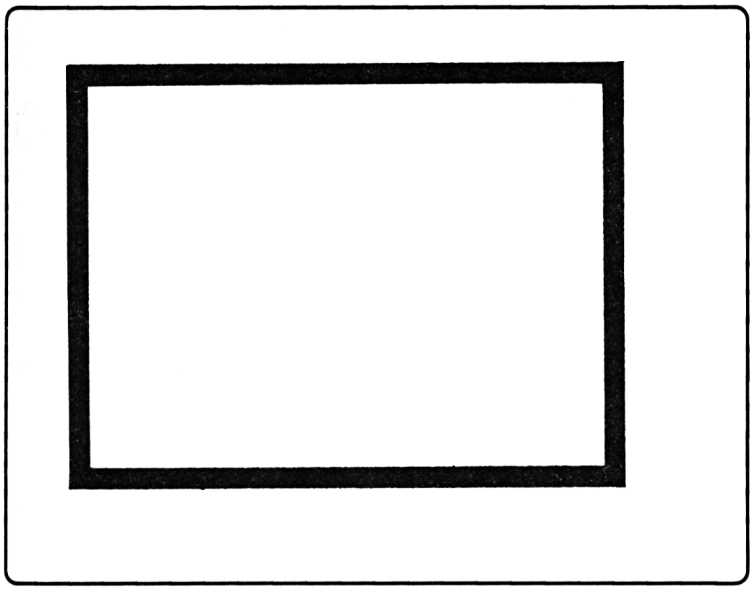
O. — Bon, alors je commence par le côté droit !... Voilà ce que donne le programme une fois modifié :

```
10 FOR C=Ø TO 28
20 PRINT AT Ø,C;"■";AT 21,C;"■
"
30 NEXT C
```

```
40 FOR L=0 TO 21
50 PRINT AT L,0;"■";AT L,28,"■
"
60 NEXT L
```

P. — Tiens, je vois que tu as dû supprimer trois colonnes au lieu de deux !...

O. — Oui, j'ai été obligé !... Avec deux colonnes, cela ne suffisait pas !... Il restait encore un espace d'un carreau en bas de chaque verticale !

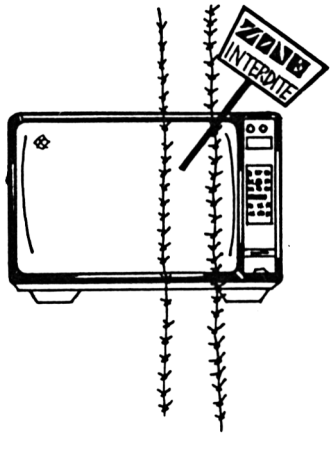


P. — Et pour le côté gauche ?

O. — Je supprime également trois colonnes, ce qui me fait débuter à la colonne numéro 3.

P. — Et qu'est-ce que tu obtiens ?

O. — La même chose !... J'ai toujours un espace de deux carreaux au pied de chaque verticale.



P. — Supprime encore quelques colonnes et tu vas comprendre pourquoi !

O. — Ok !... J'en enlève deux de plus, ce qui me fait débuter à la colonne numéro 5.

P. — Résultat ?

O. — Idem !... Attends, ce coup-ci, je mets le paquet !... Dix colonnes en moins !

P. — Et alors ?

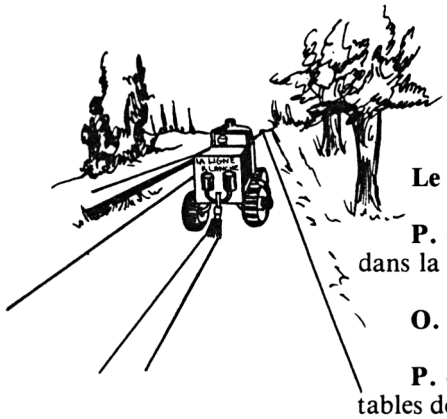
O. — Toujours pareil !... Décidément, je ne comprends pas !

P. — Tu vas comprendre si je t'explique que tu viens de déceler l'une des faiblesses de ce micro-ordinateur. Il sature très vite sa mémoire d'affichage dès que l'on travaille sur les dernières colonnes de droite.

O. — Si je te comprends bien, il vaut mieux que je m'efforce de dessiner dans la partie gauche de l'écran.

P. — Il vaut mieux en effet !

O. — Bien !... Maintenant que je sais tout cela, j'aimerais bien que tu me montres comment je peux dessiner un cadre de façon ininterrompue. Je trouve que la méthode employée n'est pas très élégante.



Le tracé continu

P. — Si je te comprends bien, tu voudrais dessiner sur l'écran comme tu le ferais dans la réalité, en parcourant le périmètre du cadre !

O. — C'est exactement cela !

P. — Bien !... Alors voilà ce que je te propose !... Etant donné les problèmes inévitables de saturation de la mémoire d'affichage que nous allons rencontrer, je te suggère de dessiner un terrain de forme carrée que tu placeras dans la partie gauche de l'écran !... Cela te convient-il ?

O. — Bonne idée !

P. — Alors, par quoi commences-tu ?

O. — Je vais démarrer par la ligne horizontale du haut, suivie par la ligne verticale de droite !

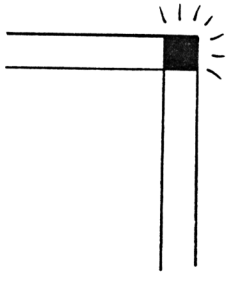
P. — Très bien !...

O. — Voilà ce que cela donne :

```
10 FOR C=0 TO 21
20 PRINT AT 0,C;"■"
30 NEXT C
40 FOR L=0 TO 21
50 PRINT AT L,21;"■"
60 NEXT L
```

P. — Tu vérifies à l'écran !

O. — C'est super !... Ça marche !



P. — Ça marche comme tu dis, mais je ne sais pas si tu t'es rendu compte que tu dessinais deux fois dans la même case !

O. — Ah bon !... Laquelle ?

P. — La dernière de la ligne horizontale et la première de la verticale !... Ce sont les mêmes !

O. — Et alors !... Je ne vois pas où est le problème !... Le carré sera tout simplement plus noir !

P. — Je suis d'accord avec toi !... Dans ce cas, il n'y en a pas !... Mais suppose que tu utilises un motif différent pour les verticales et les horizontales, il n'en sera pas de même !... Penses-y !

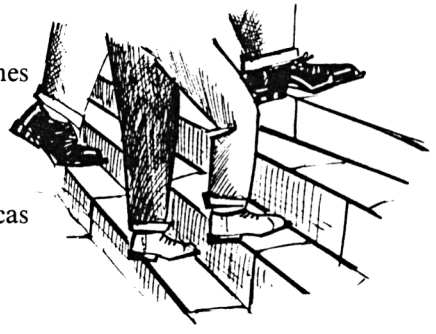
O. — Ok !... Et pour la ligne horizontale du bas, comment je fais pour dessiner en sens inverse ?

P. — Tu fais comme avant, c'est-à-dire que tu précises d'où tu veux partir et où tu veux aboutir.

O. — J'écris alors un « truc » dans le genre : FOR C = 21 TO Ø ?

P. — C'est ça !... Et tu y rajoutes l'instruction STEP – 1 afin que l'ordinateur sache qu'il faut compter à l'envers et non à l'endroit.

O. — Qu'est-ce que c'est que cette instruction STEP ?



P. — C'est une instruction qui définit le pas de progression entre les valeurs extrêmes de l'instruction FOR... TO !... D'ailleurs, « step » en anglais veut dire « pas ».

O. — Et elle est indispensable, cette nouvelle instruction ?

P. — Tout le temps, sauf lorsque le pas de progression est égal à +1. Dans ce cas seulement, on peut ne pas l'écrire.

O. — Et on peut lui donner n'importe quelle valeur au pas de progression ?

P. — Oui !... Même des valeurs décimales ou fractionnaires !

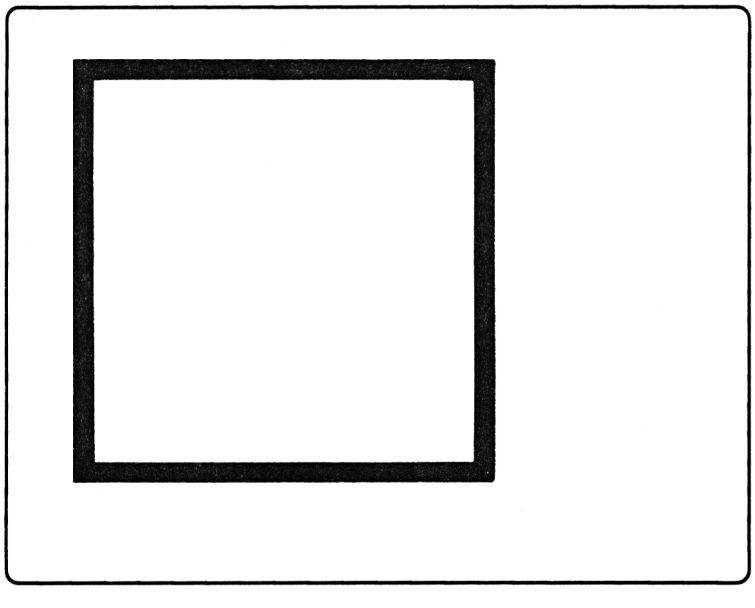
O. — Ah bon !... Donc, si j'ai bien compris, voilà ce qui devrait permettre de tracer les deux lignes manquantes :

```
 7Ø FOR C=21 TO Ø STEP -1
 8Ø PRINT AT 21,C;"■"
 9Ø NEXT C
1ØØ FOR L=21 TO Ø STEP -1
11Ø PRINT AT L,Ø;"■"
12Ø NEXT L
```

P. — Et ça marche ?

O. — Apparemment, puisque j'obtiens un beau carré qui se dessine tout seul de façon continue.

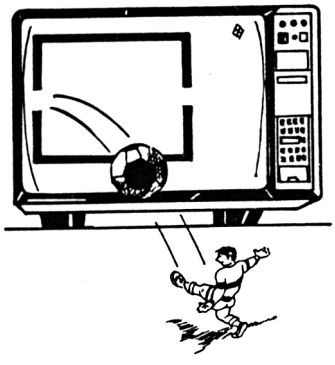
P. — Bravo !



Un terrain de football

O. — Et si maintenant je veux dessiner des buts à mon terrain de football, comment est-ce que je fais ?

P. — Ça c'est une question intéressante !... Attends que je réfléchisse un peu !... Ça y est, j'ai trouvé la solution !... Il te faut faire un saut !



O. — Un saut !... Un saut de quoi ?

P. — Un saut lors du tracé du pourtour du terrain, c'est-à-dire que tu ne dois rien dessiner à l'emplacement des buts !

O. — Ça, je m'en serais douté !... Mais comment fait-on un saut ?

P. — Il faut pour cela utiliser deux nouvelles instructions que tu ne connais pas encore. Il s'agit des instructions IF...THEN et LET.

O. — Rien que cela !

P. — Avant toute chose, il te faut d'abord dessiner exactement sur une feuille de papier quadrillé la forme de ton terrain, avec l'emplacement précis des buts.

O. — C'est déjà fait !... Je me demande toutefois si je peux utiliser toute la largeur de l'écran, étant donné les problèmes de saturation de la mémoire 1 K ?

P. — Je crois que tu dois pouvoir le faire, car ta découpe sur le côté droit te fait économiser de la place en mémoire !... D'ailleurs, quelles en sont les coordonnées ?

O. — Chaque but fait 6 carreaux de haut, entre la 8e et la 13e ligne.

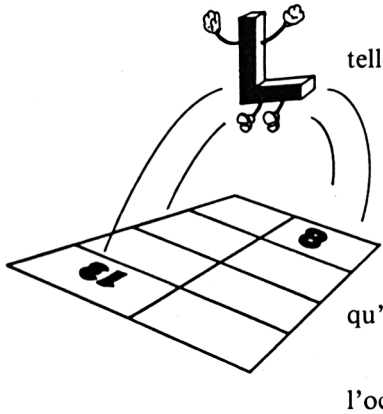
P. — Bien !... Alors, tu vas utiliser l'instruction IF pour détecter le moment où le dessin atteint la 8e ligne.

O. — IF, cela ne veut pas dire « si » en anglais ?

P. — Si, justement !... C'est une instruction qui permet de détecter une condition. Dans ton cas, si tu écris IF L = 8, cela veut dire que tu vas demander à l'ordinateur de détecter le moment où le dessin de la verticale de droite passe au niveau de la 8e ligne.

O. — D'accord !... Et ensuite ?

P. — Ensuite, tu emploies la partie THEN de cette instruction pour indiquer que tu veux qu'il se passe quelque chose lorsque cette 8e ligne est détectée. THEN correspond au mot « alors » !



O. — Si je te suis bien, cela correspond au raisonnement suivant : « si » je détecte telle condition, « alors » j'éxécute telle opération !

P. — C'est tout à fait cela !

O. — Bon, alors qu'est-ce que j'exécute ?

P. — Tu exécutes un saut de 6 carreaux en employant l'instruction LET !

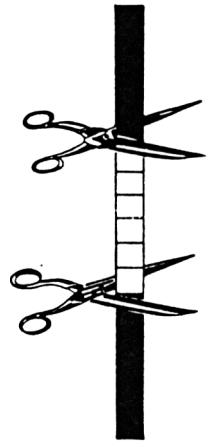
O. — Qu'est-ce que tu me gâtes avec toutes ces nouvelles instructions !... Qu'est-ce qu'elle veut dire celle-là ?

P. — C'est une instruction qui permet de définir la valeur d'une variable !... En l'occurence, il s'agit de ta variable L à qui, lorsqu'elle est égale à 8, tu fais faire un saut en lui rajoutant le nombre de carreaux de ta découpe.

O. — Ce qui, dans ce cas, correspondrait à 14 !... Mais comment dois-je écrire ce nombre après l'instruction LET ?

P. — Tout simplement comme ceci : LET L = 14 !

O. — Ah bon !... Ce n'est pas compliqué !... Maintenant, tu ne me dis plus rien. Je dois pouvoir m'en sortir tout seul !... Je vais juste tenter de dessiner une ligne verticale de toute la hauteur de l'écran avec une découpe de 6 carreaux au milieu !... Crois-tu que ce que je viens d'écrire soit correct ?



```
10 FOR L=0 TO 21
20 IF L=8 THEN LET L=14
30 PRINT AT L,31;"■"
40 NEXT L
```

P. — Tout à fait !... Je vois que tu as choisi de placer l'instruction IF...THEN au dessus de l'instruction PRINT !

O. — Ah bon !... Il y avait un moyen de faire autrement ?

P. — Oui !... Tu pouvais également la placer après l'instruction PRINT. Dans ce cas, comme tu fais faire un saut de 6 carreaux à la variable L après l'impression d'un carreau, il faut que tu commandes le saut à partir de la ligne 7 et non de la 8 comme précédemment !

O. — Deux secondes, que je vérifie !

```
10 FOR L=0 TO 21
20 PRINT AT L,31;"■"
30 IF L=7 THEN LET L=13
40 NEXT L
```

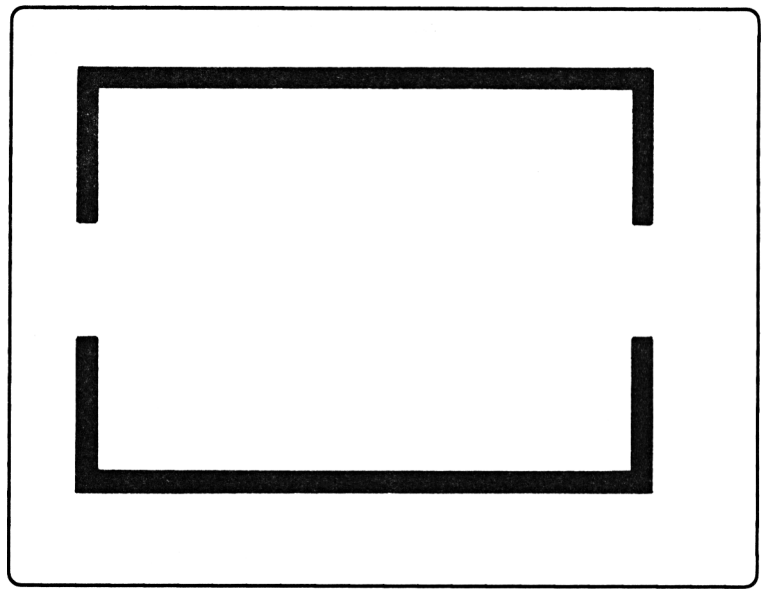
P. — Maintenant que tu as vérifié, il ne te reste plus qu'à dessiner tout entier ton terrain de football !

O. — C'est comme si c'était fait !... Que penses-tu de mon programme ?... C'est le plus long que j'ai jamais écrit !...

```
10 FOR C=0 TO 29
20 PRINT AT 0,C;"■"
30 NEXT C
40 FOR L=0 TO 21
50 IF L=8 THEN LET L=14
60 PRINT AT L,29;"■"
70 NEXT L
80 FOR C=29 TO 0 STEP -1
90 PRINT AT 21,C;"■"
100 NEXT C
110 FOR L=21 TO 0 STEP -1
120 IF L=13 THEN LET L=7
130 PRINT AT L,0;"■"
140 NEXT L
```

P. — Je note que tu as limité ton terrain à la 29e colonne !

O. — Oui, tu devines pourquoi ?



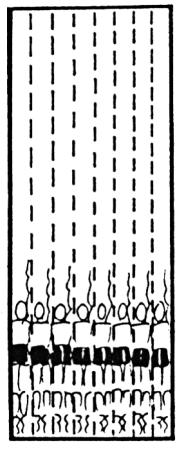
P. — Je note également que sur ta verticale de gauche tu effectues le saut à partir de la ligne 13 !

O. — Eh oui, j'ai découvert qu'il faut raisonner différemment selon que le dessin de la ligne se fait en montant ou en descendant !

P. — Mes félicitations !

Et Olivier, tout content de lui, cherchait déjà ce qu'il pourrait bien apporter comme améliorations à son terrain de football.

Une piscine de compétition



Ce n'est qu'au bout d'un quart d'heure de modifications en tous genres du terrain de football qui, dans ses derniers instants d'agonie ressemblait plus à un morceau de gruyère qu'à ce à quoi il était destiné, qu'Olivier émergea de son continuum espace-temps pour atterrir dans le mien. Il me fit aussitôt part des conclusions de ses réflexions en m'annonçant froidement :

O. — J'ai envie de dessiner une piscine de compétition !

Quel rapport pouvait-il bien y avoir entre un terrain de football et une piscine, je me le demanderais longtemps. Néanmoins, je lui répondis aimablement :

P. — Et combien de lignes d'eau faut-il à la piscine de « Monsieur » ?

O. — Ben, 8 voyons, puisque c'est une piscine de compétition !

P. — L'as-tu au moins dessinée sur une feuille de papier quadrillé ?

O. — Non, pas encore !

P. — Alors, je te conseille de le faire, car un calcul rapide me laisse entrevoir quelques petits problèmes liés au nombre de lignes d'eau que tu désires !

O. — Bien, bien !... Alors, je dessine !... Dis-donc, tu as raison !... Avec 8 lignes d'eau, j'ai besoin de 17 lignes au total, mais cela ne laisse au nageur que la largeur d'un carreau pour évoluer. C'est peu !

P. — Si tu veux élargir chaque couloir d'un carreau, il te faut dessiner la piscine dans le sens de la hauteur de l'écran.

O. — Ah ça, pas question !... Elle apparaîtrait plus large que longue.

P. — Alors, il te faut renoncer à tes huits lignes d'eau !... Tu peux déjà obtenir un résultat satisfaisant avec 6 lignes d'eau. Chaque nageur dispose alors de 2 carreaux pour progresser dans sa course.

O. — Effectivement, c'est nettement mieux !... Tu avais raison, il valait mieux faire un dessin d'abord !... Alors, par quoi on commence ?

P. — Par le cadre !

O. — Ok !... Ce coup-ci, je voudrais un cadre moins large que le précédent.

P. — Alors, je te conseille d'employer le motif en forme de rectangle. Il te donne un trait d'épaisseur moitié.

O. — Et je le dessine comment ?... En discontinu ou en continu ?

P. — En discontinu !... Le tracé continu est beau à voir, mais il est trop gourmand en mémoire.

O. — Ah oui, c'est vrai !... Toujours cette sacrée mémoire !... Bon, ça y est, j'ai fini !

```
10 FOR C=1 TO 30
20 PRINT AT 2,C;"▬";AT 20,C;"▬
"
30 NEXT C
40 FOR L=2 TO 20
50 PRINT AT L,0;"▌";AT L,31;"▌
"
60 NEXT L
```

P. — Cela me paraît très bien !... Tu l'essayes !

O. — Ok, c'est bon !... Il ne reste plus qu'à ajouter les lignes d'eau.

P. — Qu'est-ce que tu aimerais employer comme motif pour les représenter ?

O. — Je ne sais pas !... Peut-être le signe moins !

P. — Va pour le signe moins !... Il te faut donc dessiner 5 lignes d'eau espacées de 3 carreaux.

O. — Je sais quelle instruction il faut employer !... C'est STEP 3 !

P. — Exact !... Et vois-tu comment il faut procéder ?

O. — Je suppose qu'il suffit que j'utilise deux boucles emboîtées, comme on l'a fait précédemment pour les pavés noirs ?

P. — C'est cela même !... La petite boucle dessine les lignes horizontales et la grande les écarte de 3 carreaux les unes des autres.

O. — Compris !... Juste le temps de les écrire !... Voilà, c'est fait !... Qu'en penses-tu ?

P. — Encore un sacré programme, n'est-ce pas ?

```
 70 FOR L=5 TO 17 STEP 3
 80 FOR C=1 TO 30
 90 PRINT AT L,C;"-"
100 NEXT C
110 NEXT L
```

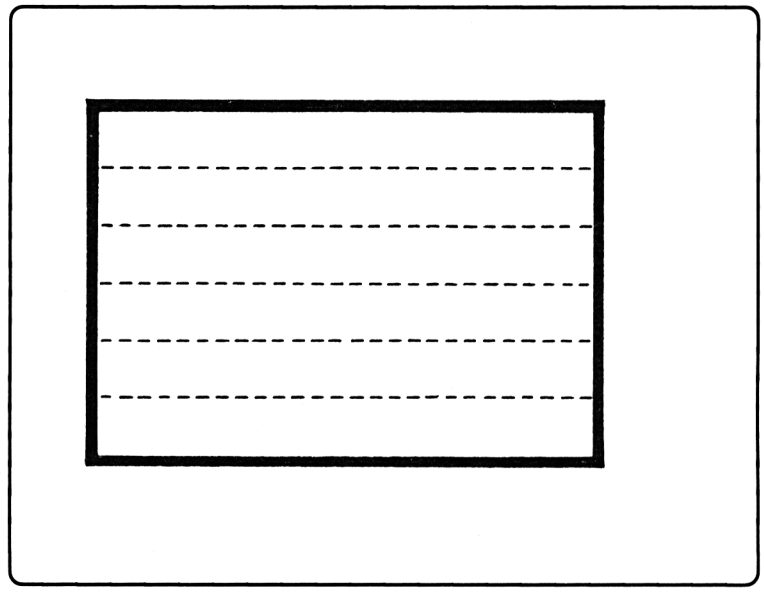
O. — Oui, mais ça ne passe pas ! J'ai une saturation de la mémoire d'affichage.

P. — Bien !... Alors il va falloir réduire un peu la longueur de la piscine.

O. — Je m'en occupe ! Voilà les corrections qu'il faut faire aux lignes 10, 50 et 80 !

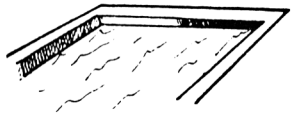
```
 10 FOR C=1 TO 26
 50 PRINT AT L,0;"▮";AT L,27;"▮
"
 80 FOR C=1 TO 26
```

P. — Et qu'est-ce que cela donne à l'écran ?



O. — C'est super chouette !

Ce qui voulait tout dire...



Une table de ping-pong

Olivier était maintenant très occupé à modifier sa piscine, tant en ce qui concerne le nombre de lignes d'eau que les dimensions de celles-ci, puisque, de toute façon, elle ne pouvait être aux normes olympiques ! Ce n'est que lorsqu'il se trouva à court d'idée que, par association avec le ping-pong qu'il pratique après la natation, il me demanda :

O. — Et si maintenant, tu me montrais comment on fait pour dessiner une table de ping-pong !

P. — Je veux bien, mais tu connais le principe !... Dessine-moi d'abord sur un papier ce que tu désires obtenir, et après je te montrerai comment y arriver !... D'autant plus que je ne connais pas les dimensions réglementaires d'une table de ping-pong !

O. — Ne t'inquiète pas pour cela, je les connais !... Il me semble qu'une table qui ferait 21 carreaux de longueur sur 12 carreaux de largeur serait « au poil » !

P. — Dans ces conditions, tu peux la placer horizontalement ou verticalement ! Quelle solution préfères-tu ?

O. — Je préfère la mettre à l'horizontale !... Comme ça, j'aurai la place pour disposer les deux joueurs de chaque côté !

P. — Je suppose que tu souhaites la centrer au milieu de l'écran ?

O. — Oui, bien sûr !... Sauf si on découvre que cela pose des problèmes de saturation de la mémoire...

P. — On verra bien, quand on y sera !... Encore un dernier point : veux-tu la table toute blanche avec un encadrement noir, ou toute noire avec un encadrement blanc ?

O. — Si cela ne complique pas trop, je préférerais que la table soit de couleur sombre, comme dans la réalité.

P. — Alors, je te suggère d'employer le programme qui t'a permis de dessiner un grand carré noir.

O. — Tu veux parler de celui qui comporte deux boucles emboîtées ?

P. — Celui-là même !

O. — Et je dessine dans quel ordre ?... Colonne par colonne ou ligne par ligne ?

P. — Etant donné que tu auras encore un filet à dessiner par la suite et qu'il te faudra le tracer verticalement, je te suggère la solution qui consiste à dessiner la table colonne par colonne.

O. — C'est vrai qu'il reste le problème du filet à solutionner !... J'avoue que d'emblée j'aurais choisi l'autre méthode !... Bon, alors comme j'ai tous les éléments en main, je programme !... Juste le temps d'écrire 5 lignes d'instructions et j'ai fini !... Voilà, c'est fait !

```
10 FOR C=5 TO 25
20 FOR L=5 TO 16
30 PRINT AT L,C;"■"
40 NEXT L
50 NEXT C
```

P. — Tu l'essayes pour voir si cela passe bien sans saturer la mémoire !

O. — Aucun problème !... Qu'est-ce que tu penses de mon idée de dessiner la table tout en noir ?

P. — Je trouve cela du meilleur effet !...Justement, puisque tu parles de motif, quel est celui que tu vas employer pour dessiner le filet ?

O. — J'avais bien pensé au signe moins, mais comme le filet est vertical, je ne peux pas l'employer !

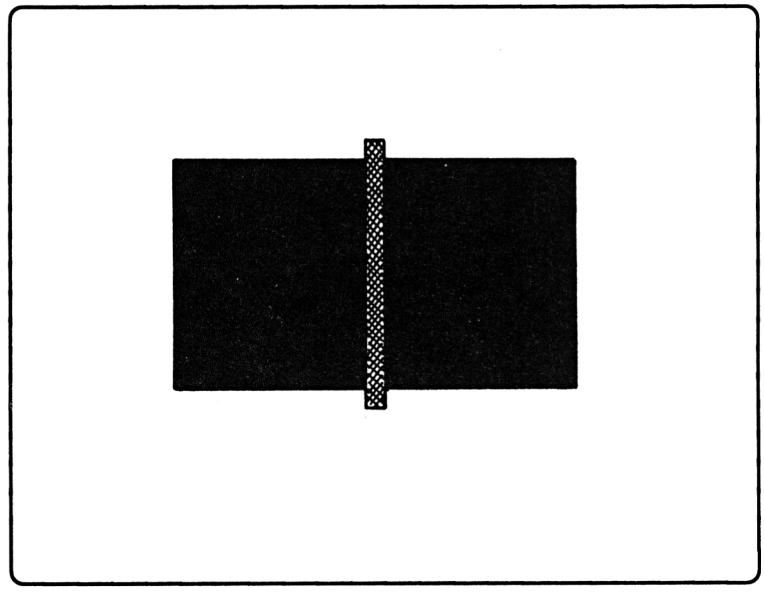
P. — Tu peux essayer les deux points ou la parenthèse !

O. — Non, cela ne me tente pas !... Tiens, je vais essayer le carré quadrillé pour voir ce que cela donne !... Par contre, je me demande comment faire pour programmer cela ?

P. — Tu as raison de te poser la question, car il existe deux méthodes. La première consiste à dessiner séparément en trois programmes distincts les deux côtés de la table et le filet, en dessinant dans l'ordre le côté gauche, puis le filet et enfin le côté droit. L'autre méthode, plus astucieuse, consiste à dessiner la table entière à l'aide d'un seul programme, comme tu l'as fait, et à remplacer ensuite sur la colonne du milieu le motif de la table par celui du filet. Avec cette seconde méthode, tu n'as plus besoin que de deux programmes au lieu de trois !

O. — Effectivement, c'est astucieux !... Alors, dans ces conditions, voici le filet :

```
60 FOR L=4 TO 17
70 PRINT AT L,15;"▒"
80 NEXT L
```



P. — Je vois que tu as choisi un filet qui déborde !

O. — C'est normal, puisque c'est comme ça dans la réalité !

P. — Si je te dis cela, c'est que si ton filet ne débordait pas de la table, il eût été possible de le dessiner avec le même programme que celui de la table, en incorporant une ligne supplémentaire du genre :

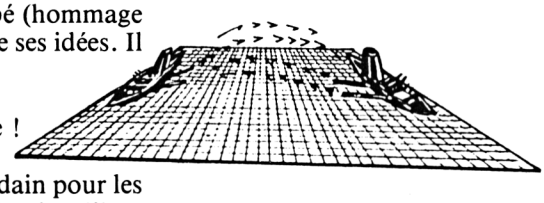
```
35 IF C=15 THEN PRINT AT L,C;"
▒"
```

P. — Cette ligne te permettrait d'économiser un peu de place en mémoire !... Mais puisque le filet doit être réglementaire...

Un quadrillage de bataille navale

Cette très intéressante construction en série de surfaces de jeux de toutes natures s'acheva sur l'appel à la soupe en provenance du monde d'en bas. La matinée entière

venait de s'écouler en un instant !... Ce n'est qu'une fois le déjeûner absorbé (hommage au passage aux talents de la cuisinière...), qu'Olivier put reprendre le cours de ses idées. Il se tourna alors vers moi et me déclara :



O. — Je dessinerais bien un quadrillage pour jouer à la bataille navale !

Je crois que je me demanderais encore pendant longtemps si ce goût soudain pour les jeux de société ne provenait pas des influences apaisantes de la digestion. Quoi qu'il en soit, le ton était donné !... Autant la matinée avait été sportive, autant l'après-midi serait intellectuelle !

P. — Donne-moi quelques indications complémentaires sur ce que tu souhaites réaliser comme quadrillage !

O. — Tout d'abord, je voudrais qu'il ait 1Ø colonnes et 1Ø lignes !

P. — Voyons cela !... Pour les colonnes, pas de problème !... Si chaque trait vertical occupe une colonne et que l'espace libre en occupe une seconde, il te faut : 2 x 1Ø + 1 = 21 colonnes au total. Cela tient aisément en largeur !... Pour les lignes, par contre, c'est plus juste !... Comme tu ne disposes que de 22 lignes et qu'il t'en faut également 21, cela ne te laisse qu'une ligne de libre. Où comptes-tu la mettre ?

O. — Tout en haut !... Comme cela, je pourrais numéroter les colonnes !

P. — Très bonne idée !... Dans le même style, je suppose que tu désires laisser une colonne libre sur la gauche du quadrillage pour numéroter les lignes ?

O. — Oui, bien sûr !

P. — Et quel motif vas-tu employer ?

O. — J'avoue ne pas savoir !... Qu'est-ce que tu me conseilles ?

P. — Ce qui est certain, c'est que le motif que tu vas employer devra présenter le même aspect, qu'il soit dessiné horizontalement ou verticalement !

O. — Ça ne me laisse pas beaucoup de choix !... Je vois le signe plus, l'étoile, le carré noir et le carré hachuré.

P. — Il reste aussi les deux damiers que tu oublies !

O. — C'est vrai !... Allez, je choisis le signe plus !

P. — Dans ces conditions, il te faut procéder en deux étapes. Tu vas d'abord dessiner tes lignes horizontales en les espaçant d'un carreau, et tu feras ensuite de même pour les lignes verticales.

O. — Tout ceci ne me paraît pas trop difficile !... Pour les lignes horizontales, il suffit que j'emploie le même programme que celui qui m'a servi pour les lignes d'eau de la piscine, c'est-à-dire :

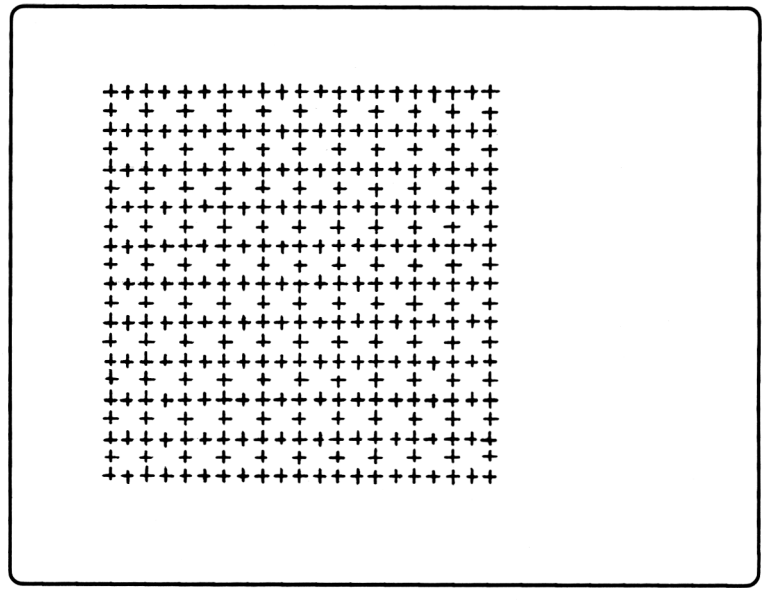
```
10 FOR L=1 TO 21 STEP 2
20 FOR C=1 TO 21
30 PRINT AT L,C;"+"
40 NEXT C
50 NEXT L
```

P. — C'est exactement cela !... Et pour les lignes verticales, que fais-tu ?

O. — J'inverse tout bonnement mes lignes et mes colonnes, ce qui donne :

```
 60 FOR C=1 TO 21 STEP 2
 70 FOR L=1 TO 21
 80 PRINT AT L,C;"+"
 90 NEXT L
100 NEXT C
```

P. — Amuse-toi à vérifier ce que cela donne à l'écran, je t'expliquerai quelque chose plus tard !



Et l'écran nous renvoya aussitôt l'image tant attendue d'un quadrillage se dessinant d'abord ligne par ligne, puis ensuite colonnes par colonnes. Satisfait, Olivier me questionna :

O. — Qu'est-ce que tu voulais m'expliquer ?

P. — Sais-tu que tu peux simplifier considérablement l'écriture de ton programme de quadrillage ?

O. — Ah bon !... Et comment ?

P. — C'est toi qui m'en as donné l'idée quand, pour écrire ton second programme, tu m'as dit qu'il suffisait d'inverser les lignes et les colonnes.

O. — Et alors ?

P. — Alors, si tu examines ton second programme et que tu y intervertisses les variables L et C, tu retrouves exactement ton premier programme, à l'exception de l'instruction PRINT AT dont les coordonnées sont inversées !

O. — Tiens, c'est vrai !

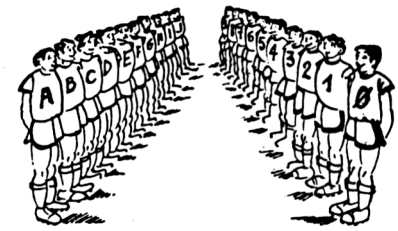
P. — Il te suffit donc d'inclure cette ligne d'instruction dans ton premier programme, en inversant la position des variables L et C, et le tour est joué !

O. — Comme çà ?

```
 35 PRINT AT C,L;"+"
```

P. — Absolument !...

O. — Whaooo, quelle astuce !... Et le pire, c'est que ça marche !



L'affichage des coordonnées

P. — Que dirais-tu si on s'occupait maintenant des coordonnées !

O. — Ah, c'est vrai !... Je les avais oubliées celles-là !

P. — J'imagine que tu veux désigner les 1Ø colonnes par les lettres de A à J, et les lignes par les chiffres de 1 à 1Ø.

O. — Oui !... Même que je préfèrerais faire l'inverse, c'est-à-dire numéroter les colonnes de 1 à 1Ø et désigner les lignes avec les lettres de A à J !

P. — Alors, si tu veux bien, on va examiner tour à tour chaque possibilité, en commençant par les lettres à l'horizontale et les chiffres à la verticale.

O. — Attends, ne dis plus rien !... Pour inscrire les lettres de A à J sur la ligne du haut, il suffit que j'écrive l'instruction PRINT AT, suivie des 1Ø lettres en question.

P. — En n'oubliant pas de laisser un espace entre chaque lettre, de façon à ce qu'elles tombent en face de chaque colonne !

O. — C'est vrai, j'oubliais !... Bon, et bien voilà le travail :

```
60 PRINT AT Ø,2;"A B C D E F G
H I J"
```

P. — Cela me paraît tout à fait correct !

O. — Maintenant que c'est fait pour les lettres, passons aux chiffres !...

P. — Vois-tu comment procéder ?

O. — Pas tout à fait !... J'avais imaginé pouvoir utiliser une boucle pour dessiner des motifs verticaux. L'ennui, c'est que le motif est à chaque fois différent, puisqu'il s'agit d'un chiffre !

P. — Alors, je vais satisfaire ta curiosité !... Tu as raison quand tu dis qu'il faut employer une boucle !... Simplement, il faut l'écrire un peu différemment de ce à quoi tu es habitué.

```
70 FOR L=1 TO 10
80 PRINT AT 2*L,Ø;L
90 NEXT L
```

O. — Hou, la, la !... C'est plein de nouveautés ton truc !... J'en vois au moins deux dans l'instruction PRINT !

P. — Que je t'explique !... Comme il faut inscrire les chiffres de 1 à 1Ø uniquement sur les lignes paires, il faut donc progresser de deux en deux. C'est pourquoi tu vas trouver le signe * entre le chiffre 2 et la variable L. Ce signe correspond à l'opération de multiplication.

O. — Alors, 2 * L, cela veut dire « deux que multiplie L » ?

P. — Exactement !

O. — Il faut le savoir !... Et le signe de la division, qu'est-ce que c'est ?

P. — C'est la barre oblique que tu trouves sur la touche [V].

O. — Ah bon !... Mais, pour en revenir à la progression de deux en deux, tu aurais pu employer l'instruction STEP 2.

P. — Effectivement, j'aurais pu !... Mais tu vas voir pourquoi j'ai choisi de ne pas le faire !... N'as-tu pas remarqué qu'après le point virgule de l'instruction PRINT AT, la lettre L n'est pas entre guillemets.

O. — Si !... Je pensais même qu'il s'agissait d'une erreur !

P. — En fait, cela veut dire que tu demandes à l'ordinateur d'imprimer non pas la lettre L mais la valeur en chiffre qu'elle représente !

O. — C'est pratique ça dis donc !

P. — Enormément !... Tu verras que nous nous en resservirons souvent par la suite !

O. — Ok, je note !... Je vérifie à l'écran ce que cela donne !... Tiens, le nombre 1Ø pénètre dans le quadrillage !

P. — C'est normal, puisque c'est le seul nombre à deux chiffres !... Pour pallier à cela, je ne vois que la solution qui consiste à numéroter les lignes de Ø à 9 !

O. — Ou bien de prévoir deux colonnes pour la numérotation des lignes !... Comme je n'ai pas envie de changer le programme de mon quadrillage, je vais modifier à ma façon le programme d'écriture des coordonnées des lignes !

P. — Fais comme il te plaît !... Je parie toutefois que tu vas employer l'instruction STEP 2 !

O. — Gagné !... Et je vais devoir diviser par deux la valeur de L après le point virgule, je sais !... Tiens, que penses-tu de cela ?

```
70 FOR L=Ø TO 18 STEP 2
80 PRINT AT L+2,Ø;L/2
90 NEXT L
```

P. — La même chose que l'ordinateur !... Appuie sur [RUN] et sur [NEW LINE] et tu le sauras !

```
  A B C D E F G H I J
0
1
2
3
4
5
6
7
8
9
```

O. — C'est tout bon !... Et maintenant, tu m'expliques comment disposer les lignes à la place des colonnes, et vice versa ?

P. — D'accord !... Comme tu t'en doutes, l'écriture horizontale des chiffres ne pose aucun problème. Seule l'écriture verticale des lettres de A à J en pose un !

O. — Effectivement, je ne vois pas comment on peut faire !

P. — Il existe heureusement une solution qui passe par l'emploi d'une nouvelle instruction : l'instruction CHR$!

O. — Tu ne veux pas répéter, s'il te plaît !

P. — C... H... R... dollar !... Le dernier caractère, qui est un S barré, correspond au symbole du dollar !... N'oublie pas que le langage BASIC est d'origine américaine !...

O. — Et à quoi sert cette instruction ?

P. — Elle te permet d'obtenir un des 256 caractères, instructions ou motifs graphiques que ton ordinateur possède en mémoire, si tu connais son numéro de code.

O. — Tu veux dire que ce petit ordinateur possède 256 symboles différents ? C'est pas croyable !

P. — Tu en trouveras la liste à la page 181 de ton manuel d'instruction. La première colonne indique le numéro de code de chaque symbole !

O. — Ainsi, si je t'ai bien compris, si je tape CHR$ 2Ø, j'obtiendrais le signe = ?

P. — Oui, mais à condition de l'écrire après l'instruction PRINT !... Sinon tu n'obtiendras rien !

O. — Bien sûr, c'est ce que je voulais dire !

P. — Sur cette liste, tu remarqueras entre autre que les chiffres de Ø à 9 ont pour code les nombres 28 à 37, et que les lettres de A à Z en vidéo normale ont pour code les nombres de 38 à 63.

O. — Et que les chiffres en vidéo inversée ont les nombres de 156 à 165 pour code !... C'est génial !

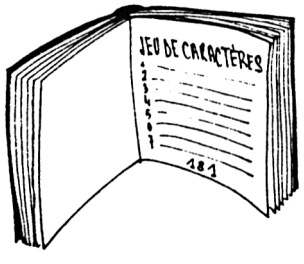
P. — Maintenant que tu sais tout, il ne te reste plus qu'à t'exécuter !

O. — Ok, c'est parti !

P. — Bien que cela ne soit pas la façon de procéder la plus simple, ni la plus économique, je te suggère de réécrire en entier ton programme d'affichage des coordonnées, en te servant de cette nouvelle instruction CHR$!

O. — Cela va me prendre un peu plus de temps que d'habitude, car c'est assez laborieux à calculer !... Voilà, c'est fini !

```
 6Ø FOR C=28 TO 37
 7Ø PRINT AT Ø,2*(C-27);CHR$ C
 8Ø NEXT C
 9Ø FOR L=38 TO 47
1ØØ PRINT AT 2*(L-37),Ø;CHR$ L
11Ø NEXT L
```



P. — Et ça marche !... Bravo !... Et en vidéo inversée, tu ne veux pas essayer ?

O. — Si !... Justement, je cherche dans le manuel d'utilisation les numéros de code du chiffre Ø et de la lettre A !

Un damier d'échecs

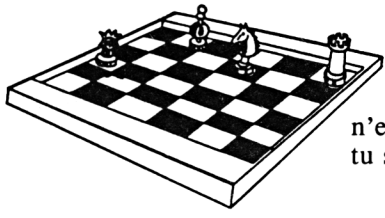
Je laissais donc Olivier modifier tout seul son programme pour obtenir l'affichage des coordonnées en vidéo inversée. Cela ne le passionna pas longtemps puisque, très peu de temps après, il m'appela pour me dire :

O. — Si on s'attaquait maintenant à un damier de jeu d'échecs !

P. — Bonne idée !... Mais uniquement le damier !... Pour la tactique du jeu, il te faudra t'adresser à quelqu'un d'autre !

O. — Oui, oui, ne t'inquiète pas !

P. — Bien !... Alors, dans ces conditions, il ne nous reste plus qu'à trouver une solution pour dessiner un damier de 8 x 8 cases, en alternant les cases noires et blanches.



O. — Ça ne me paraît pas évident du tout, du tout !

P. — Et pourtant, quand je t'aurai expliqué comment procéder, tu verras que ce n'est guère plus compliqué que ce que tu viens de réaliser jusqu'à présent !... La preuve : tu sais déjà dessiner une case noire de plusieurs carreaux de côté ?

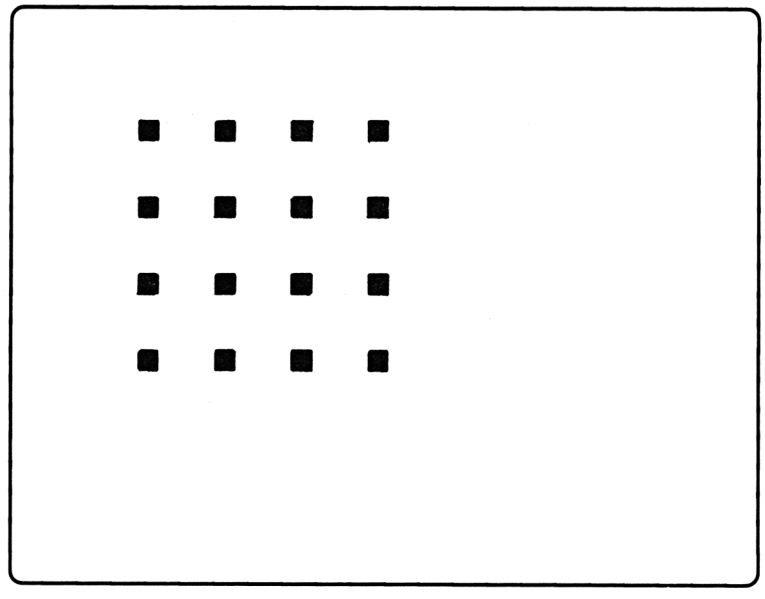
O. — Oui !... Il suffit d'employer deux boucles emboîtées !

P. — Bien !... Tu sais aussi, j'imagine, dessiner un quadrillage de points sur l'écran, en les espaçant à volonté d'un nombre déterminé de carreaux ?

O. — Pas précisément !

P. — Alors, je vais te montrer !... Suppose que tu veuilles dessiner une matrice de 4 x 4 points écartés de 4 carreaux !... Il te suffit d'écrire :

```
10 FOR C=3 TO 15 STEP 4
20 FOR L=3 TO 15 STEP 4
30 PRINT AT L,C;"■"
40 NEXT L
50 NEXT C
```

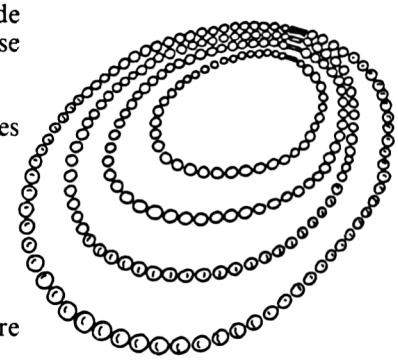


O. — J'ai déjà écrit cela plusieurs fois sans obtenir cette matrice de points !... Comment as-tu fait ?

P. — J'ai ajouté l'instruction STEP 4 après chaque instruction FOR...TO, alors que, jusqu'à présent, tu ne l'avais employée qu'une seule fois !

O. — Et c'est ça qui fait la différence ?... Que c'est plein d'astuces la programmation !

P. — Pour en revenir au damier, il te suffit donc d'écrire le programme précédent qui te permet d'obtenir une matrice de points, et de remplacer l'instruction PRINT AT de la ligne 30 qui n'imprime qu'un seul carré noir par le programme de dessin d'une case noire de plusieurs carreaux de côté !



O. — Ok, j'ai compris !... Mais dis donc, au total cela représente quatre boucles emboîtées !

P. — Et cela te fait peur ?

O. — Tu plaisantes, j'espère !

P. — Une dernière remarque, avant que tu ne te lances à corps perdu dans l'écriture de ton programme !... Quelles vont être les dimensions des cases de ton damier ?

O. — Deux carreaux par deux carreaux ! Je ne peux les faire plus grandes, sinon cela ne tiendrait pas en hauteur sur l'écran !

P. — Et que placeras-tu plus tard sur ces cases ?

O. — Des pièces d'échecs, voyons !

P. — Je suppose que tu symboliseras alors d'une façon ou d'une autre chaque pièce par un caractère ?

O. — Oui, bien sûr !

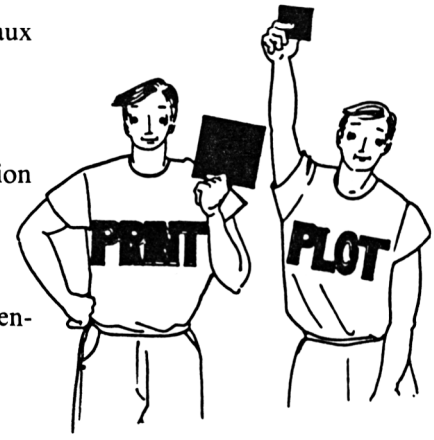
P. — Et où les disposeras-tu ces caractères ?

O. — Ben, au centre de chaque case !... Ça y est, je viens de comprendre pourquoi tu me poses toutes ces questions !... Il n'est pas possible d'écrire au milieu d'une case constituée de 2 x 2 carreaux !... Mince alors !... Comment faire ?

P. — Il faudrait que tu dessines chaque case en empiétant sur la moitié des carreaux qui entourent le carreau central !

O. — D'accord, mais c'est trop difficile à faire !

P. — Avec l'instruction PRINT, je suis d'accord !... Mais connais-tu l'instruction PLOT ?



O. — L'instruction PLOT ?... Qu'est-ce que c'est que ça ?

P. — C'est une instruction qui permet de dessiner des petits carrés noirs de dimension moitié de ceux de l'instruction PRINT !

O. — Très intéressant, dis donc !

P. — Je te crois !... D'autant plus que, comme cette instruction ne peut dessiner autre chose que des petits carrés noirs, son écriture en est simplifiée !

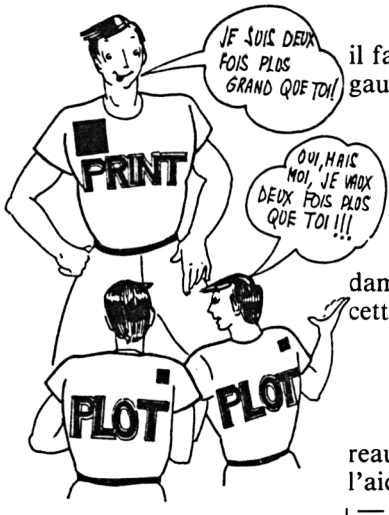
O. — Encore plus intéressant !... Continue !

P. — Elle s'écrit tout simplement : PLOT (colonne), (ligne) et c'est tout !

O. — Tiens, les coordonnées des lignes et des colonnes sont inversées par rapport à l'instruction PRINT !

P. — Exact !... De surcroît, l'écran n'est plus divisé en 32 colonnes et 22 lignes, mais en 64 demi-colonnes et 44 demi-lignes !... Le double, quoi !

O. — Et comment sont numérotées ces demi-lignes et ces demi-colonnes ?



P. — De Ø à 63 pour les demi-colonnes et de Ø à 43 pour les demi-lignes. Par contre, il faut que tu fasses attention à l'origine (Ø, Ø) des coordonnées. Elle se situe dans le coin gauche en bas de l'écran.

O. — Comme sur un graphe ?

P. — Exactement !

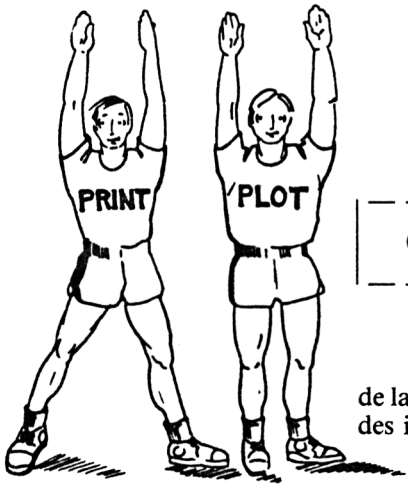
O. — En clair, si je comprends bien, je vais pouvoir dessiner chaque case noire du damier en employant quatre demi-colonnes de quatre demi-carreaux de hauteur, grâce à cette instruction PLOT !

P. — Eh oui !

O. — Et il ne me reste plus qu'à me débrouiller pour les centrer autour d'un carreau !... Alors, dis-moi ce que tu penses de ce petit programme de dessin d'une case à l'aide de l'instruction PLOT :

```
10 FOR I=5 TO 8
20 FOR J=5 TO 8
30 PLOT I,J
40 NEXT J
50 NEXT I
```

P. — Pas mal !... Pour être bien certain du centrage de ta case, je te conseille de dessiner dessus le symbole d'une pièce d'échecs.



O. — Ça, c'est une bonne idée !... Voyons, qu'est-ce que je vais choisir ?

P. — Ce que tu veux !... Pourquoi pas la lettre R pour le roi !

O. — Comme ça ?

```
60 PRINT AT 18,3;"R"
```

P. — Oui, tout à fait !... Ainsi, comme tu peux le voir, la lettre R apparaît au centre de la case !... Maintenant, je voudrais profiter de cet exercice pour te faire remarquer l'un des inconvénients majeurs de cette instruction PLOT !

O. — Je le devine facilement !... Avec tout le mal que j'ai eu pour déterminer les coordonnées de la lettre R !

P. — C'est exactement cela !... Les coordonnées de cette instruction ne correspondent pas avec celles de l'instruction PRINT. Si pour la colonne, il suffit de diviser par deux la coordonnée de l'instruction PLOT pour tomber sur celle de l'instruction PRINT, à condition d'arrondir, l'inverse n'est vrai qu'un coup sur deux. Quant aux lignes, c'est une autre paire de manches, puisqu'il faut soustraire de 43 les coordonnées de ligne de l'instruction PLOT et diviser ensuite le résultat par deux.

O. — Tu parles d'une gymnastique !

P. — C'est pourquoi il faudra t'efforcer, lorsque tu concevras un jeu vidéo, à ne pas mélanger les deux instructions PRINT et PLOT !... De toute façon, on en reparlera le moment venu !

O. — Compris !... Pour en revenir au damier, il me semble que j'ai tous les éléments en main pour pouvoir le dessiner tout seul !

P. — Alors, à toi de jouer !... Par quoi commences-tu ?

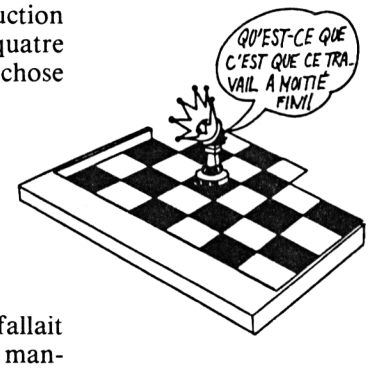
O. — Par le programme de la grille de points dans lequel je vais remplacer le dessin d'un point par le programme de dessin d'une case !... Voilà ce que cela donne :

```
10 FOR X=5 TO 29 STEP 8
20 FOR Y=5 TO 29 STEP 8
30 FOR I=0 TO 3
40 FOR J=0 TO 3
50 PLOT X+I,Y+J
60 NEXT J
70 NEXT I
80 NEXT Y
90 NEXT X
```

P. — Et que penses-tu de ce que tu obtiens à l'écran ?

O. — J'en pense qu'il me manque la moitié des cases !... Comment puis-je faire pour obtenir l'autre moitié ?

P. — Je vais te proposer une solution très simple !... Ajoute une ligne d'instruction qui va t'imprimer en même temps que chaque case une seconde case décalée de quatre demi-carreaux en hauteur et en largeur par rapport à la première !... Quelque chose comme :




```
55 PLOT X+I+4,Y+J+4
```

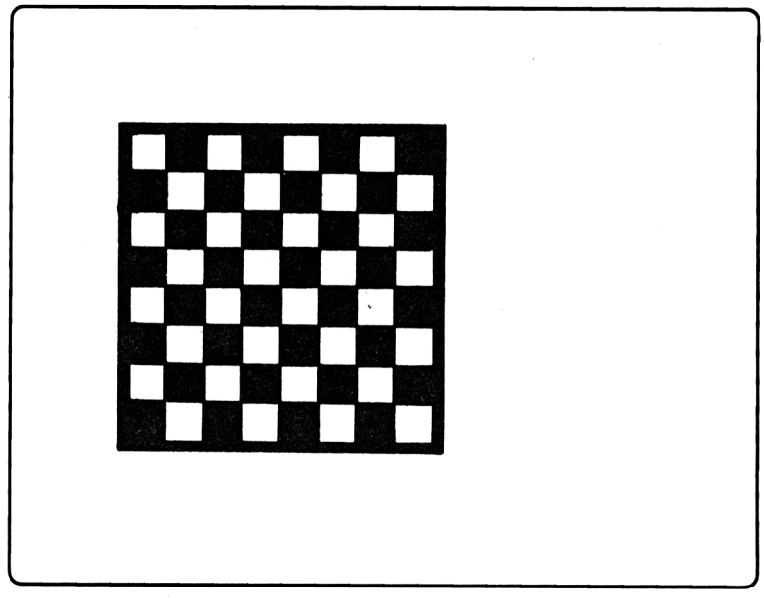
O. — T'as gagné, c'est bon !... Quand je pense qu'à un moment, j'ai cru qu'il fallait écrire tout un second programme aussi long que le premier pour obtenir les cases manquantes !

P. — Et pour finir en beauté ton damier, tu ne veux pas y dessiner tout autour une bande noire ?

O. — Si, bien sûr !... Ça c'est facile, même en instruction PLOT !

```
100 FOR C=4 TO 37
110 PLOT C,4
120 PLOT C,37
130 NEXT C
140 FOR L=4 TO 37
150 PLOT 4,L
160 PLOT 37,L
170 NEXT L
```

Et un superbe cadre se dessina autour du damier, achevant ainsi sa réalisation. Olivier était en extase devant son œuvre, clamant bien fort et bien haut que l'informatique était un « truc » génial !... Je le ramenai à notre problème, car il n'était pas complètement terminé.



P. — Ton programme de cadre est splendide, mais maintenant que tu as progressé, je vais t'enseigner une astuce pour en raccourcir l'écriture !

O. — Encore une !... Vas-y, je t'écoute !

P. — Si tu regardes ton programme de près, tu peux te rendre compte qu'il comporte deux boucles dans lesquelles les deux variables C et L ont même valeur. Dans ces conditions, il te suffit de les regrouper en une seule et même boucle, comme cela :

```
100 FOR P=4 TO 37
110 PLOT P,4
120 PLOT P,37
130 PLOT 4,P
140 PLOT 37,P
150 NEXT P
```

P. — ...et ainsi, tu économises deux lignes d'instructions devenues inutiles !

O. — Intéressant !... Et pourquoi avoir choisi la variable P ?

P. — J'ai pris P comme « pourtour » ou « périmètre », car je ne voulais pas employer en tant que variable unique les lettres C et L que je réserve aux colonnes et aux lignes.

O. — Ah bon !...

Un court de tennis

L'après-midi avançait gaiement et , après avoir terminé le dessin du damier, je me demandais bien vers quoi l'imagination d'Olivier irait se fixer. La réponse vint sous la forme d'un appel en provenance de l'étage inférieur où les fervents du tennis, que sont aussi les autres membres de la famille, avaient branché le téléviseur pour assister à la finale de Roland-Garros entre Noah et Wilander.

Nous nous y précipitâmes en courant, avides que nous étions de ne pas manquer une seule minute de cette partie qui s'annonçait superbe. Néanmoins, au bout de quelques minutes, je vis Olivier ressortir en coup de vent de la pièce et y revenir équipé de plusieurs feuilles de papier et de tout le matériel nécessaire pour dessiner. Je le questionnais :

P. — Qu'est-ce que tu vas faire avec tout cela ?

O. — Chut !... C'est pour dessiner un court de tennis !

Cette opération s'avéra laborieuse et grande consommatrice de papier, car elle dura autant que la partie elle-même. Ce ne fut que lorsque les esprits furent apaisés par la victoire de Noah qu'Olivier m'annonça :

O. — Si j'arrive à dessiner à l'écran la même chose que sur mon papier, j'avoue que je ne serais pas mécontent de moi !

P. — Fais voir ton dessin !... Pas mal, dis donc !... Les proportions sont bien respectées !... Cela va constituer un excellent exercice d'emploi de l'instruction PLOT, car je vois que tu as tout dessiné avec des traits d'un demi-carreau d'épaisseur !

O. — Sauf le filet bien sûr, pour lequel j'ai choisi d'utiliser le carré hachuré !

P. — Ce qui me paraît être du meilleur effet !

O. — Bien !... Alors j'espère que tu vas trouver mon programme aussi bien que mon dessin !... En attendant la suite, voilà déjà pour les deux couloirs :

```
10 FOR I=9 TO 52
20 PLOT I,8
30 PLOT I,12
40 PLOT I,32
50 PLOT I,36
60 NEXT I
```

P. — Pas mal !... Je remarque que tu en as profité pour centrer ton court sur l'écran !

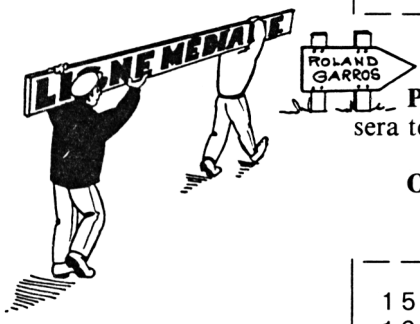
O. — C'est normal !... Il me fallait de la place libre de chaque côté pour disposer plus tard les joueurs !... Alors, puisque cela te plaît, je continue par les deux lignes de fond de court qui vont délimiter la surface de jeu !... Le plus dur avec cette instruction PLOT, c'est de déterminer les coordonnées des demi-lignes et des demi-colonnes !

```
70 FOR J=8 TO 36
80 PLOT 9,J
90 PLOT 52,J
100 NEXT J
```

P. — Pas vilain du tout !... J'imagine maintenant que tu vas continuer par les deux lignes de service ?

O. — Tu as gagné !... Juste le temps de trouver leurs coordonnées exactes et les voici :

```
110 FOR J=12 TO 32
120 PLOT 19,J
130 PLOT 42,J
140 NEXT J
```



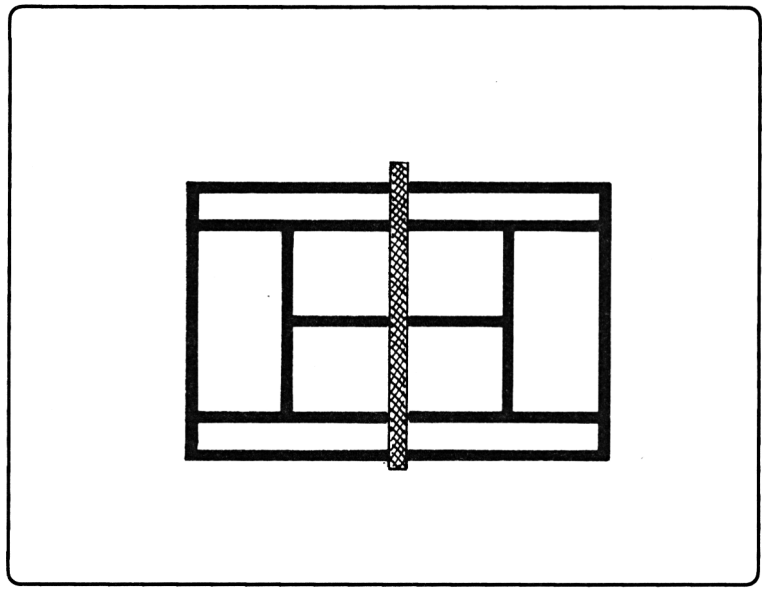
P. — Ça devient bon !... Plus que la ligne médiane et le filet à dessiner, et ton court sera terminé !

O. — Alors, voilà pour la ligne médiane :

```
150 FOR I=19 TO 42
160 PLOT I,22
170 NEXT I
```

O. — ... et voici pour le filet :

```
180 FOR L=3 TO 18
190 PRINT AT L,15;"▒"
200 NEXT L
```



Et un magnifique court de tennis s'étala devant nos yeux, court capable de nous faire vivre des moments aussi intenses que ceux de la finale.

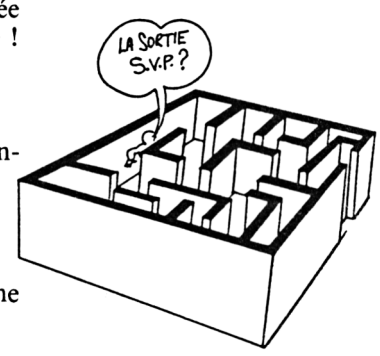
Un labyrinthe

L'épisode tennis étant terminé, il ne restait plus pour finir cette seconde journée d'informatique qu'à conclure par ce qu'Olivier me réclamait depuis le petit matin, à savoir un labyrinthe pour pouvoir jouer au PAC-MAN. Je lui proposais donc :

P. — Alors, on se le fait ce labyrinthe ?

O. — Oh oui !... Chouette alors !

P. — Tu vas très vite comprendre pourquoi j'ai attendu jusqu'à la fin de la journée pour te le montrer !... Je te préviens, ce n'est pas ce que l'on pourrait qualifier de facile !



O. — Je m'en « fiche » !... Allez, explique !

P. — D'abord, as-tu dessiné ton labyrinthe sur papier ?... Tu sais que c'est indispensable !

O. — Non, pas encore !

P. — Alors, c'est par là qu'il faut commencer !... Quand tu auras décidé de la forme qu'il doit avoir, tu comprendras mieux les explications que je te donnerai !

O. — D'accord, d'accord, j'ai compris !... Dis voir, pour les couloirs, je laisse combien de carreaux ?

P. — Trois !... Pour des raisons que je t'exposerai plus tard, il te sera plus facile de te déplacer au sein de ton labyrinthe si tu circules au milieu du couloir avec un espace blanc de protection situé de part et d'autre !

O. — Alors, voilà le labyrinthe !... Tu remarqueras que je l'ai commencé à la colonne Ø pour le terminer à la colonne 2Ø, ce qui représente 5 couloirs verticaux de trois carreaux chacun, et qu'il va de la ligne Ø à la ligne 2Ø, ce qui représente également 5 couloirs horizontaux de trois carreaux chacun.

P. — Très bien !... Puisque tout est prêt, on va commencer par dessiner les cloisons intérieures.

O. — J'avoue que je ne vois pas comment faire !

P. — Ne t'inquiète pas !... La première des choses qu'il faut remarquer, et c'est là où le dessin est indispensable pour comprendre, c'est qu'il existe des points particuliers qui peuvent te servir de référence pour dessiner tes cloisons !

O. — Tu veux parler des carrés noirs situés aux points d'intersection des lignes verticales avec les horizontales sur lesquelles j'ai prévu de construire les cloisons ?

P. — Exactement !... Si tu examines bien ces points, tu vas constater qu'ils sont espacés de quatre en quatre et que, quelle que soit l'orientation des cloisons intérieures, ils demeurent toujours noirs !

O. — C'est vrai !... Et alors ?

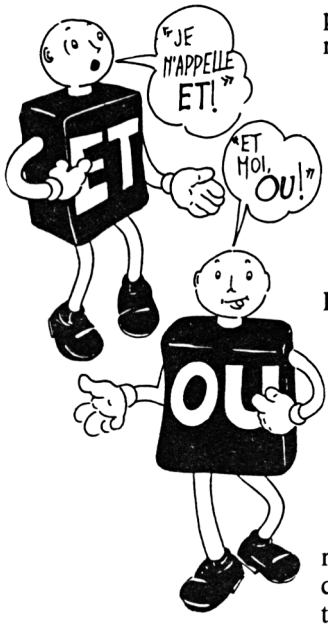
P. — Alors il te suffit d'écrire le programme capable de tracer une grille de points espacés de quatre en quatre !... Je sais que tu sais le faire, puisque tu l'as déjà employée par le damier !

O. — Tu veux parler de cela ?

```
10 FOR C=Ø TO 2Ø STEP 4
2Ø FOR L=Ø TO 2Ø STEP 4
3Ø PRINT AT L,C;"■"
4Ø NEXT L
5Ø NEXT C
```

P. — Précisément !... Je remarque au passage que ton labyrinthe aura une forme carrée !... Maintenant, tu vas remplacer la ligne 3Ø qui ne convient pas par deux lignes d'instruction , chacune d'entre elles étant chargée de déterminer s'il convient de dessiner une cloison horizontale ou verticale !

O. — Précise un peu ta pensée car, pour le moment, je ne comprends rien !



P. — Voilà !... Lorsque le programme d'exploration des points arrive sur un point particulier, il faut lui dire s'il faut dessiner une cloison horizontale ou verticale de 5 carreaux.

O. — Et comment fait-on cela ?

P. — A l'aide des instructions logiques AND et OR !

O. — C'est quoi ça ?

P. — L'instruction AND est une instruction qui correspond à la fonction logique ET, c'est à dire à la fonction d'intersection entre deux ensembles.

O. — Et à quoi sert-elle ?

P. — Elle sert à déterminer une condition liée à un couple de valeurs !

O. — Désolé, mais je n'ai toujours pas compris !

P. — Attends, je vais prendre un exemple !... Je vais d'ailleurs choisir celui du labyrinthe, précisément !... Suppose que tu veuilles dessiner une cloison horizontale à partir du point de coordonnées ligne 8, colonne 4. Il faut donc signaler par une instruction que tu veux que cela se fasse lorsque la variable de la ligne vaut 8 **ET** que celle de la colonne vaut 4 !

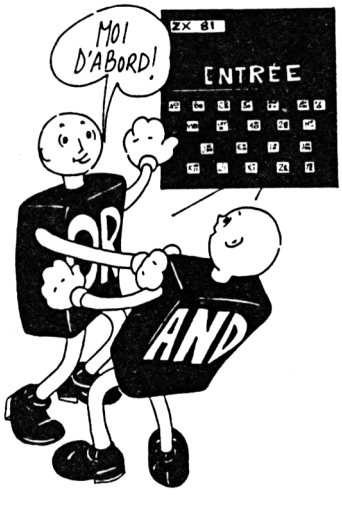
O. — Ça y est, j'ai compris !... J'imagine que cela doit s'écrire : IF L = 8 AND C = 4 THEN etc...

P. — C'est tout à fait cela !

O. — Et pour l'instruction OR ?

P. — Il s'agit dans ce cas de l'instruction qui correspond à la fonction logique OU, c'est-à-dire à la fonction d'union entre deux ensembles.

O. — Donne-moi un exemple d'utilisation de cette instruction OR !



P. — Ok !... On va reprendre l'exemple précédent !... Suppose cette fois que tu veuilles dessiner une seconde cloison horizontale à partir du point de coordonnées 12, Ø. Il te faut donc indiquer à l'ordinateur que tu attends qu'il fasse cela lorsque le point de coordonnées 8, 4 est atteint **OU** lorsque c'est celui de coordonnées 12, Ø.

O. — Ce qui s'écrirait je suppose : IF L = 8 AND C = 4 OR L = 12 AND C = Ø THEN etc...

P. — C'est tout à fait cela !... Ah ! tant que j'y pense, il faut que tu saches que lorsque tu emploies simultanément les deux instructions AND et OR, c'est l'instruction OR qui est exécutée avant l'instruction AND !

O. — Et ces instructions sont limitées à deux ensembles seulement ?

P. — Non, il peut y en avoir autant que tu le souhaites. Il suffit pour cela de répéter l'instruction concernée autant de fois que c'est nécessaire !

O. — Bien, je tâcherai de m'en souvenir !... Pour en revenir maintenant au labyrinthe, j'ai compris les explications concernant les coordonnées, mais ce que je ne vois pas, c'est comment je dois procéder pour tracer une cloison horizontale plutôt qu'une cloison verticale, ou vice versa !

P. — La difficulté est double !... Elle provient d'abord du fait qu'il y a deux tracés différents, c'est-à-dire qu'il faut écrire deux programmes distincts, un par type de cloison. La seconde difficulté réside dans le choix qu'il faut faire entre ces deux programmes !... Heureusement, il existe une solution à cela !... Elle consiste à utiliser ce qui s'appelle un sous-programme !

O. — Un sous-programme ?... Qu'est-ce que c'est ?

P. — C'est un petit programme que l'on a écrit et placé dans un coin à part, et auquel on peut faire appel en cas de besoin à l'aide de l'instruction GOSUB.

O. — GOSUB ?... Est-ce que par hasard, cela ne voudrait pas dire en anglais : « va au sous-programme » ?

P. — Si !... Quand un ordinateur rencontre cette instruction dans un programme, il exécute aussitôt le sous-programme prévu avant de retourner au programme principal, au point où il avait été interrompu.

O. — Il faut donc indiquer à l'ordinateur le numéro de la ligne d'instruction où débute le sous-programme !

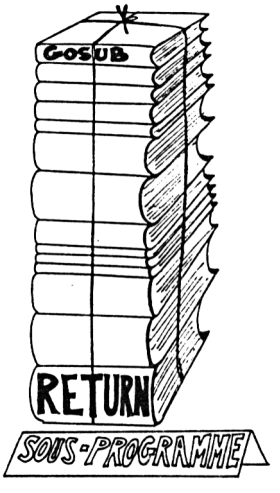
P. — Exact !... Cela s'écrit en faisant suivre l'instruction GOSUB du numéro de la première ligne du sous-programme, comme par exemple : GOSUB 2ØØ !

O. — Et comment l'ordinateur peut savoir quand un sous-programme est terminé ?

P. — Tout simplement parce que tu le lui précises à l'aide d'une nouvelle instruction qui s'appelle RETURN et que tu places en fin de sous-programme.

O. — Alors, si je comprends bien, il n'y a pas intérêt à ce que j'oublie d'écrire cette instruction RETURN, sinon l'ordinateur s'arrête tout seul à la dernière ligne du sous-programme et ne reprend plus le déroulement du programme là où il l'avait interrompu.

P. — Eh oui !... Maintenant que nous avons en main toutes les données nécessaires, il ne nous reste plus qu'à écrire le programme du labyrinthe !... Voilà ce que je te propose : j'écris la partie consacrée aux cloisons horizontales et toi, tu fais celle des cloisons verticales !



O. — Ok !... A toi de commencer !

P. — Bien !... Si tu te souviens, il fallait remplacer la ligne 3Ø, qui ne convenait pas, par deux lignes d'instructions, une par type de cloison !... D'après ton dessin, il y a 7 cloisons horizontales à dessiner. Voici donc la ligne d'appel du sous-programme :

```
25 IF L=12 AND C=Ø OR L=8 AND
C=4 OR L=16 AND C=4 OR L=12 AND
C=8 OR L=4 AND C=12 OR L=12 AND
C=12 OR L=8 AND C=16 THEN GOSUB
2ØØ
```

O. — J'y retrouve les 7 paires de coordonnées qui indiquent à partir de quels points il convient de dessiner une cloison horizontale.

P. — C'est cela !... Au passage, remarque le choix du numéro de la première ligne du sous-programme : 2ØØ. Ainsi, tu es sûr qu'avec un tel numéro, il se retrouvera loin après la fin du programme principal, et ne se mélangera pas avec lui.

O. — C'est vrai qu'il faut faire attention à cela !

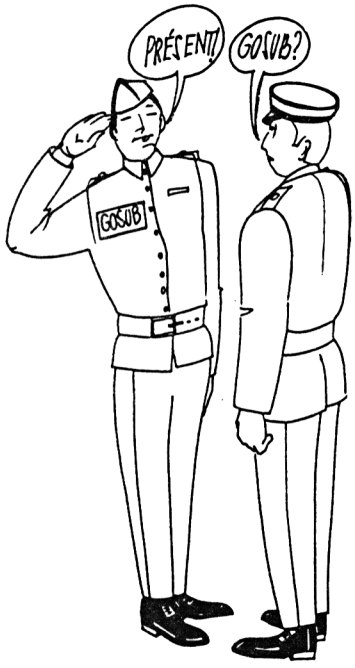
P. — Après l'instruction d'appel du sous-programme, voici maintenant le sous-programme en question :

```
2ØØ FOR A=Ø TO 4
21Ø PRINT AT L,C+A;"■"
22Ø NEXT A
23Ø RETURN
```

O. — Dans le fond, il n'y a que l'instruction RETURN qui permette de faire la différence entre un sous-programme et un programme normal !

P. — C'est vrai !... Maintenant, c'est à toi de jouer !

O. — Bien !... Il faut donc que je dessine 5 cloisons verticales !... Voici donc l'instruction d'appel du sous-programme :



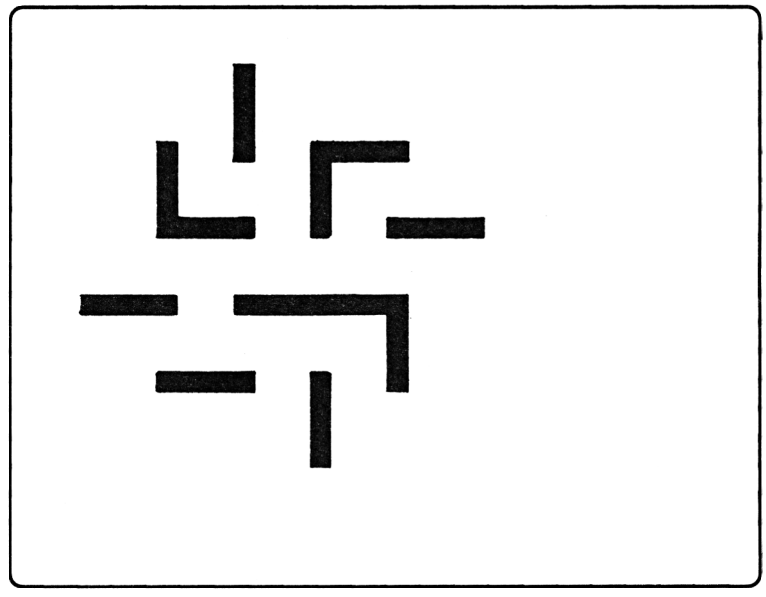

```
35 IF L=4 AND C=4 OR L=0 AND C
=8 OR L=4 AND C=12 OR L=16 AND C
=12 OR L=12 AND C=16 THEN GOSUB
300
```

O. — ... et voici le sous-programme en question :

```
300 FOR A=0 TO 4
310 PRINT AT L+A,C;"■"
320 NEXT A
330 RETURN
```

P. — Bravo !... Tu appuies sur [RUN] et [NEW LINE], qu'on voit ce que cela donne à l'écran.

Et l'ordinateur dessina les cloisons intérieures du labyrinthe, les unes après les autres, en procédant de gauche à droite...



P. — Remarquable, n'est-ce pas ?

O. — Fan-tas-ti-que !... Quand tu penses que c'est moi qui ai réalisé cela !

P. — Une belle illustration de la prédominance de l'esprit sur la matière !... Pour en revenir à des considérations moins philosophiques, il nous reste encore à tracer le cadre autour du labyrinthe.

O. — Laisse-moi faire ! Ça, c'est plus facile à écrire que le programme précédent !

```
60 FOR P=0 TO 20
70 PRINT AT 0,P;"■";AT 20,P;"■
";AT P,0;"■";AT P,20;"■"
80 NEXT P
```

P. — Je vois avec plaisir que tu as bien assimilé l'écriture simplifiée du tracé d'un cadre carré !... Bravo !... Fais voir ce que cela donne à l'écran !

O. — Tiens, ça ne passe pas !... J'obtiens un message d'indication d'une saturation de la mémoire d'affichage.

P. — En fait, ce coup-ci, ce n'est pas seulement la mémoire d'affichage qui est saturée, c'est également la mémoire de programmation !

O. — Tu veux dire que mon programme est trop long ?

P. — Non, pas vraiment !... Simplement, il gaspille beaucoup d'espace en mémoire de la façon dont il est écrit !

O. — Ah bon !... Je l'ai pourtant écrit comme tu me l'as indiqué !

P. — Bien sûr, puisque je voulais justement t'amener à toucher du doigt ce problème délicat de saturation de la mémoire de programmation !

O. — Et de quoi cela provient-il ?

P. — D'une utilisation abusive des chiffres !... Ce qu'il te faut savoir, c'est que, chaque fois que tu frappes un nombre, l'ordinateur, qui ne sait pas à l'avance combien de chiffres il comporte, lui réserve systématiquement huit emplacements de mémoire.

O. — Tu parles d'un gaspillage !... Et comment peut-on remédier à cela ?

P. — En remplaçant les nombres par des lettres, car une lettre n'occupe qu'un seul emplacement de mémoire !... Ce qu'il faut faire dans ce cas, c'est affecter une valeur particulière à chaque lettre que tu emploies.

O. — Là, je te vois venir !... Tu vas me sortir une nouvelle instruction capable de faire tout ça !

P. — Tu as bien deviné !... Il s'agit de l'instruction LET dont la syntaxe exacte d'écriture veut que tu fasses suivre cette instruction de la lettre et de la valeur que tu veux lui affecter, comme par exemple : LET A = 2 !

O. — Et si je comprends bien, une fois que j'aurais affecté une valeur à une lettre, il me suffit d'employer cette lettre à la place du nombre que j'aurais écrit pour obtenir le même résultat !

P. — C'est tout à fait cela !... Personnellement, j'utilise un code de correspondance tout simple entre les lettres et leur valeur. Je remplace le chiffre Ø par la lettre Z, le 1 par U, le 2 par D, le 4 par Q, etc... Comme cela, c'est plus facile à s'en souvenir !

O. — Alors, il ne me reste plus qu'à tout réécrire en employant cette fois des lettres à la place des chiffres !... Où me conseilles-tu de placer les différentes instructions LET ?

P. — Combien t'en faut-il ?

O. — Il m'en faut six !... Une pour le Ø, le 4, le 8, le 12, le 16 et le 2Ø !

P. — Dans ces conditions, je te conseille de les disposer en début de programme, avant ta première ligne d'instruction !

O. — Voilà, c'est fait !... Qu'en penses-tu ?

```
1 LET Z=Ø
2 LET Q=4
3 LET H=8
4 LET D=12
5 LET S=16
6 LET V=2Ø
```

P. — C'est parfait !... Malgré cela, je vais te faire recommencer, car tu peux encore économiser de l'espace en mémoire en remplaçant tous les nombres, sauf un, par des lettres.

O. — Alors là, je ne comprends plus !

P. — Attends, je vais te montrer ce que je veux dire :

```
1 LET Q=4
2 LET Z=Q-Q
3 LET H=Q+Q
4 LET D=H+Q
5 LET S=H+H
6 LET V=S+Q
```

O. — J'ai compris !... C'est génial ton « truc » !... Il suffit d'utiliser une seule fois le chiffre 4 !... Bon, et bien je vais réécrire le reste du programme !... Qu'est-ce que c'est long !... Voilà, j'ai terminé :

```
 1Ø FOR C=Z TO V STEP Q
 2Ø FOR L=Z TO V STEP Q
 3Ø IF L=D AND C=Z OR L=H AND C
=Q OR L=S AND C=Q OR L=D AND C=H
 OR L=Q AND C=D OR L=D AND C=D O
R L=H AND C=S THEN GOSUB 2ØØ
 4Ø IF L=Q AND C=Q OR L=Z AND C
=H OR L=Q AND C=D OR L=S AND C=D
 OR L=D AND C=S THEN GOSUB 3ØØ
 5Ø NEXT L
 6Ø NEXT C
2ØØ FOR A=Z TO Q
21Ø PRINT AT L,C+A;"■"
22Ø NEXT A
23Ø RETURN
3ØØ FOR A=Z TO Q
31Ø PRINT AT L+A,C;"■"
32Ø NEXT A
33Ø RETURN
```

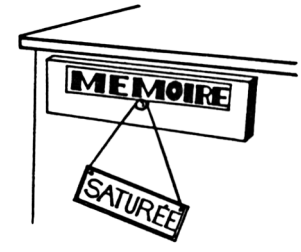
P. — Tu vérifies si cela marche !

O. — Oui, c'est bon !

P. — Alors maintenant, tu peux ajouter ton programme de dessin du cadre extérieur.

O. — C'est pratiquement terminé !... Voilà la chose :

```
70 FOR P=Z TO V
80 PRINT AT Z,P;"■";AT V,P;"■"
;AT P,Z;"■";AT P,V;"■"
90 NEXT P
```



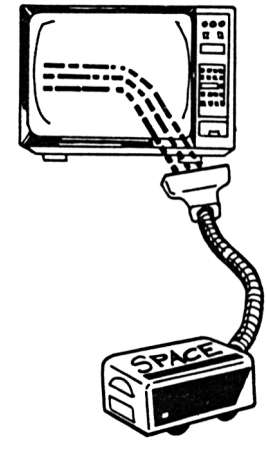
P. — Et qu'est-ce que tu obtiens à l'écran ?

O. — Un tout petit bout du cadre !... La mémoire est à nouveau saturée !

P. — Alors, je vais t'enseigner une dernière astuce pour gagner encore de la place !... Pour cela, il te faut supprimer ton programme de dessin du cadre extérieur, et le remplacer par les instructions suivantes que tu incorpores dans les lignes 30 et 40 selon qu'il s'agit des lignes horizontales ou verticales :

```
30 IF L=Z AND C<=S OR L=V AND
C<=S OR..........
40 IF L<=S AND C=Z OR L<=S AND
 C=V OR.............
```

O. — J'ai compris le truc !... Tu profites du programme de dessin des cloisons pour dessiner le cadre extérieur !... Je suppose que cette astuce économise pas mal d'espace en mémoire ?

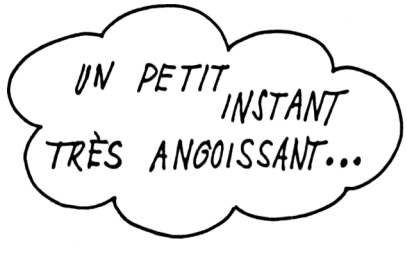


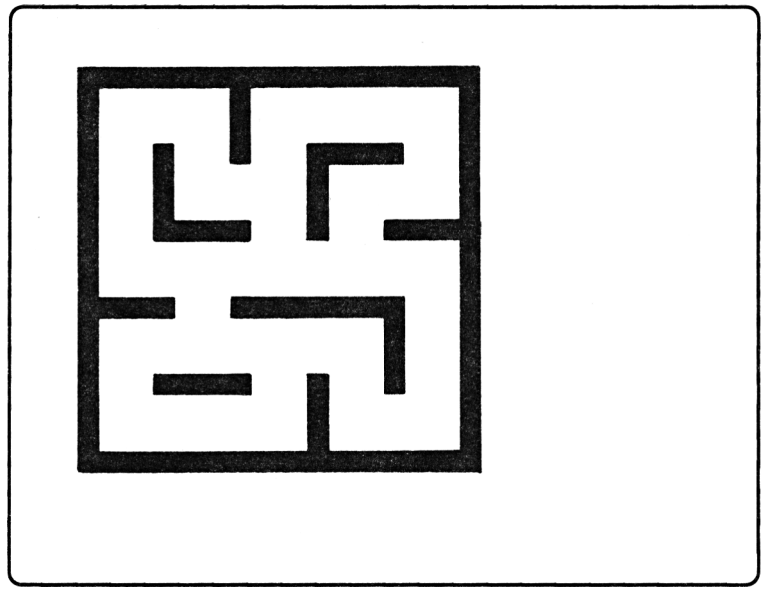
P. — Tu ne le sauras qu'en faisant l'essai !

O. — Il faut donc que j'ajoute ces quatre paires de coordonnées dans les lignes 30 et 40 !... Allons-y avec la première de ces lignes !... Tiens, je n'arrive pas à éditer la ligne 30 ! J'appuie pourtant sur [SHIFT] et sur [EDIT] !... Qu'est-ce que cela veut dire ?

P. — Ce que tu constates se produit chaque fois que la mémoire de programmation est proche de la saturation !... Lorsque cela se produit, il faut appuyer à plusieurs reprises sur la touche SPACE, jusqu'à ce que le curseur, à force de se déplacer de gauche à droite, efface suffisamment d'instructions pour faire de la place pour la ligne que tu veux éditer !

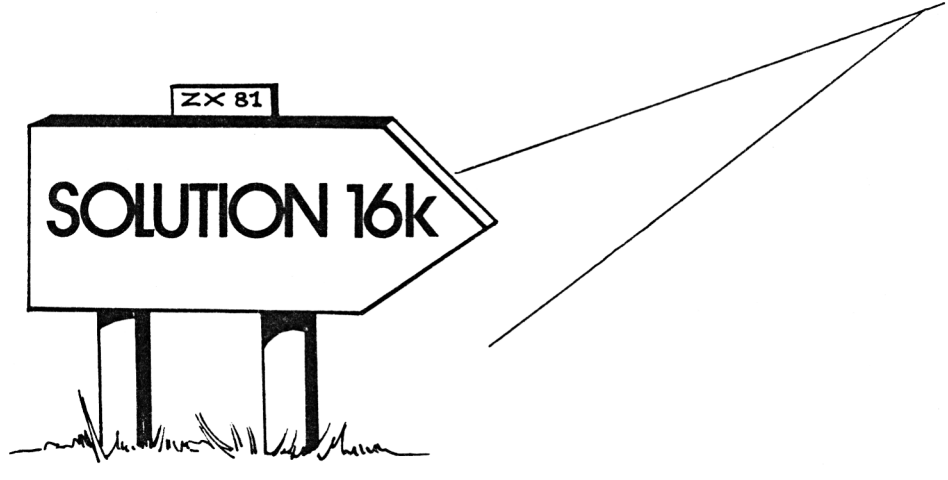
O. — Ok, j'ai compris !... Ça y est, ça marche !... Voilà, j'ai fini !... J'appuie sur [RUN] et [NEW LINE] !...





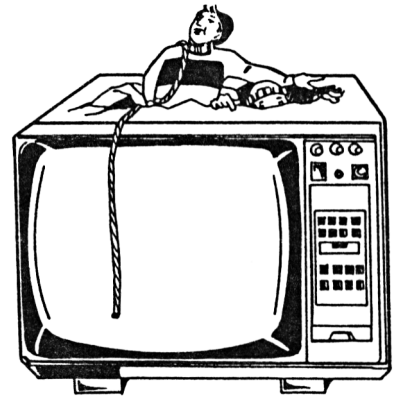
O. — Whaooo, c'est fantastique !... Mince, il me manque un bout de la verticale gauche !

P. — Moi, je trouve que ce n'est pas mal !... Pour ce qui manque, il ne reste plus que la solution de la mémoire 16 K !



ENGINS ET DECORS

C'est complètement exténués qu'Olivier et moi, nous terminâmes cette seconde journée d'informatique sur la réalisation du labyrinthe. Qu'est-ce qu'il portait bien son nom celui-là !... Il illustrait très exactement ce que fut notre démarche tout au long de cette journée. Nous n'arrêtions pas de nous enfoncer plus profondément à chaque nouvelle réalisation, incapables d'imaginer qu'il puisse exister une porte de sortie à notre progression.



Or, c'est justement l'insuffisance de capacité de la mémoire de 1 Kmot qui nous a indiqué comment il était possible d'en sortir et de remonter à la surface. C'est beau la spéléologie des profondeurs des jeux vidéo, mais il faut de temps en temps revenir à la réalité quotidienne !... Olivier me confirma qu'il effectuait une démarche similaire quand, au cours du repas du soir, il me déclara :

O. — Dis donc Papa, dans tout ça, on n'a pas eu le temps de dessiner le moindre engin !

P. — C'est vrai !... On était parti pour cela et, de terrain de football en court de tennis, on a dévié de notre objectif !

O. — Il ne nous reste plus qu'à remettre ça à samedi prochain !

Et voilà, c'en était fait de mes samedis !... Quand je racontais au début de cet ouvrage que la méthode de travail émergea très vite de nos tâtonnements, cela fut notamment le cas pour la détermination du jour idéal que l'on consacrerait à l'informatique !... Ce serait désormais tous les samedis !... L'avantage toutefois d'une telle formule, c'est qu'elle laissait une semaine de répit. Je décidai de profiter lâchement de cette situation en conseillant à Olivier :

P. — En attendant samedi prochain, amuse-toi donc à imaginer quelques-uns des engins que tu souhaiterais voir apparaître sur ton écran !

O. — Ça, c'est une idée !... Comme ça, on gagnera du temps !

Et c'est effectivement comme cela que la semaine se passa. Je voyais la progression d'Olivier dans le dessin de ses engins, ne serait-ce qu'au nombre impressionnant de papiers qu'il laissait traîner un peu partout et qu'il ne fallait surtout pas jeter !... C'est ainsi que le samedi suivant, je vis arriver mon Olivier avec une pile incroyable de petits bouts de papier destinés, à n'en pas douter, à alimenter cette troisième journée consacrée aux jeux vidéo. Je lui demandais :

P. — Alors, la pêche a été bonne ?

O. — Pas trop mauvaise !... J'ai mis au point une soucoupe, deux ou trois avions, un fusée, la navette spatiale...

P. — Ça, c'est le salon du Bourget qui t'a inspiré !

O. — Bien sûr !... Je continue : une voiture de course, un super train, un sous-marin, quelques bateaux dont un porte-avions, et sûrement d'autres que j'oublie !

P. — Bravo, dis donc !... Moi, de mon côté, je ne suis pas resté inactif !... Je t'ai imaginé quelques décors dont tu me donneras des nouvelles !

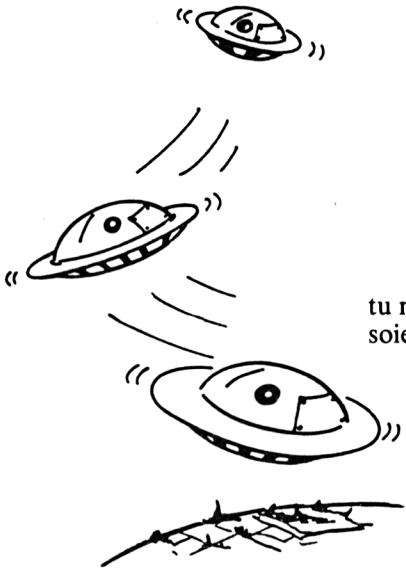
O. — Ah oui, lesquels ?

P. — Tu les verras en temps utile !... Il doit y avoir une ville, un parcours de golf, une grotte souterraine et bien d'autres choses encore...

O. — Chouette programme tout ça !... Par quoi on commence ?

P. — Par ce que tu as fait !... D'après ce que tu m'as dit, il me semble qu'on pourrait peut-être débuter par les engins qui volent.

O. — Alors, allons-y pour ma soucoupe volante !



Une soucoupe volante

P. — Fais voir un peu le dessin de cette soucoupe !

O. — Tiens !... J'espère qu'elle te plaît ?

P. — Elle est superbe !... Le seul problème, c'est qu'elle est trop grande !... Comme tu ne disposes que de très peu de lignes et de colonnes sur ton écran, il faut que tes engins soient tout petits, de façon qu'ils puissent évoluer assez librement !

O. — Et à ton avis, quelle est la taille idéale que devrait avoir ma soucoupe ?

P. — De deux à trois carreaux de longueur et pas plus de deux carreaux de hauteur !

O. — C'est peu !

P. — Oui, bien sûr, mais c'est nécessaire !... De surcroît, tu découvriras que lorsque tu mettras tes engins en mouvement, il ne faudra pas qu'ils soient trop grands, sinon ils donneraient l'impression d'avancer morceau par morceau, comme s'ils étaient constitués de pièces détachées.

O. — Alors, il ne me reste plus qu'à m'exécuter et à réduire la taille de ma soucoupe !... Voilà, c'est fait !... Maintenant, elle ne mesure plus que trois carreaux par un !

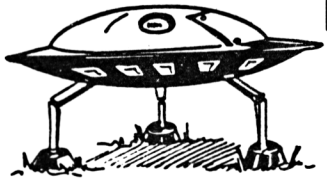
P. — Ça, c'est nettement mieux comme taille !

O. — Tu veux la voir ?

P. — Bien sûr !... Tu n'as qu'à la placer à peu près au milieu de l'écran !

O. — Aussitôt dit, aussitôt fait !... Et une soucoupe qui marche... pardon, qui vole..., une !

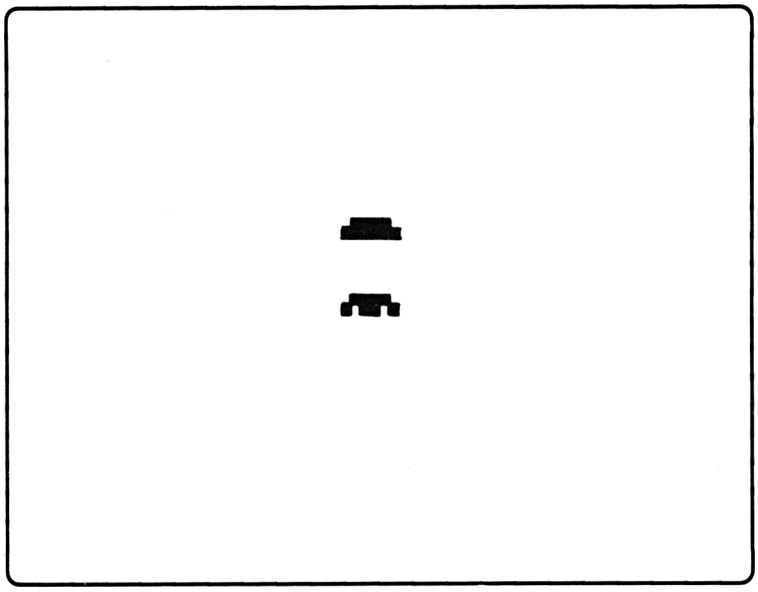
```
10 PRINT AT 8,14;"▟▙"
```



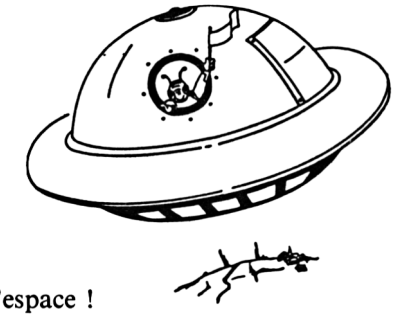
P. — Elle est magnifique, dis donc !... On la croirait réelle !... Maintenant, c'est à mon tour de t'en proposer une autre :

```
20 PRINT AT 12,14;"▗▟▙▖"
```

O. — Eh, la tienne n'est pas mal non plus !... Elle fait encore plus vraie avec son tripode d'atterrissage !



P. — Bien !... Puisque nous sommes en si bonnes dispositions de créativité, toi et moi, je te propose de continuer par quelques véhicules extra-terrestres !



Quelques véhicules extra-terrestres

O. — Très bonne idée !... Comme ça, je pourrai jouer aux envahisseurs de l'espace !

P. — Ce n'est pas tout à fait ainsi que je voyais les choses !... Quand je t'ai parlé de véhicules extra-terrestres, je ne sous-entendais pas qu'ils puissent servir à envahir notre planète. Je ne pense pas qu'il faille nécessairement considérer les extra-terrestres comme des montres assoiffés de conquêtes. Ils sont peut-être plus sages que nous le sommes...

O. — Bon, ça va, j'ai compris la leçon !... Je jouerai donc à autre chose avec, voilà tout ... Bien, qui de nous deux commence ?

P. — A toi l'honneur !

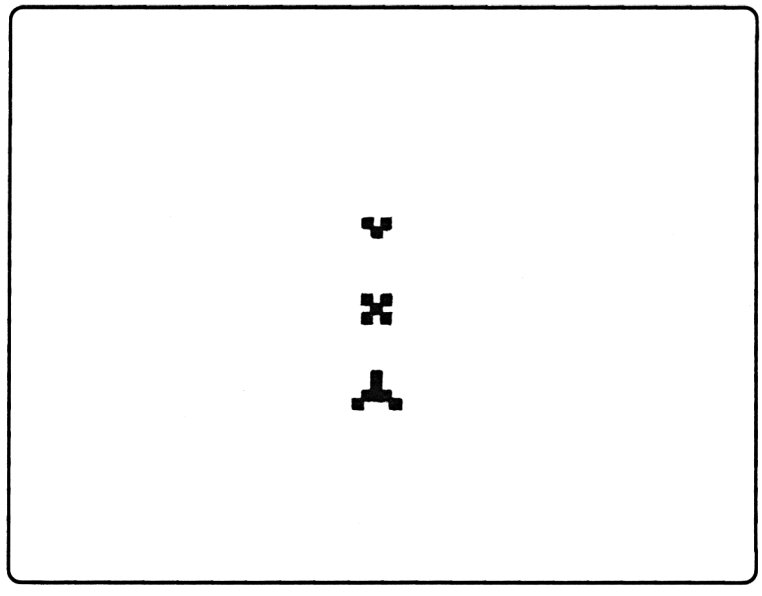
O. — Alors, voici le programme du premier véhicule :

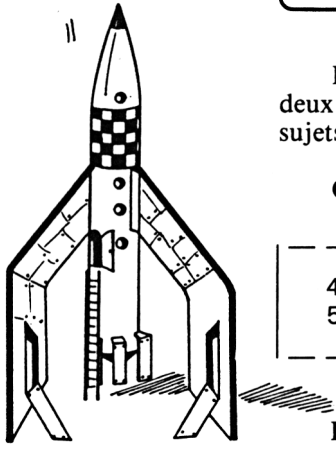
```
10 PRINT AT 8,15;"▚▞ "
```

P. — Il est mignon tout plein !... On dirait une sorte d'aile volante !... Cela me donne d'ailleurs une idée pour le suivant !... Qu'est-ce que tu penses de cela ?

```
20 PRINT AT 12,15;"▚▞ "
30 PRINT AT 13,15;"■■ "
```

O. — Pas mal !... Je remarque qu'il t'a fallu deux lignes pour l'écrire !





P. — Oui, et c'est un maximum !... Par la suite, il ne faudra jamais dépasser plus de deux carreaux en hauteur sauf, bien sûr, s'il s'agit d'engins immobiles, ou en tout cas, peu sujets aux grands déplacements.

O. — Comme cette fusée par exemple :

```
40 PRINT AT 16,15;"█"
50 PRINT AT 17,14;"▄▀▄"
```

P. — Superbe !... On dirait qu'elle va décoller de sa base de lancement !...

O. — Il y a toutefois une question que je me pose : vaut-il mieux dessiner ces petits engins avec l'instruction PRINT ou avec l'instruction PLOT !

P. — Avec l'instruction PRINT !... Si tu regardes l'exemple précédent de l'aile volante qui ne comportait que trois petits carrés noirs, il ne t'avait fallu qu'une seule instruction PRINT pour l'écrire, alors qu'il te faudrait trois instructions PLOT pour obtenir le même résultat. Toujours l'économie !... Et encore, c'est le plus petit engin que tu aies dessiné !... Tu imagines ce que cela donnerait avec la fusée...

La navette spatiale et des avions

O. — Ok !... Maintenant que je sais comment il faut s'y prendre pour dessiner correctement les engins volants, je vais te proposer ma navette spatiale, tu sais celle que j'ai vu passer dans le ciel pendant les demi-finales de tennis à Roland-Garros !

P. — Fais-moi voir un peu ça !

```
10 PRINT AT 8,10;"▪"
20 PRINT AT 9,10;"▀▄▄"
```

P. — ... Ta navette présente effectivement quelques liens de parenté avec la vraie !... Non, je te blague !... Elle est magnifique, tu sais !... Tu me donnes d'ailleurs une idée : si j'allonge le fuselage de ta navette d'un carreau, et si j'y rajoute des hublots, je devrais obtenir un avion cargo :

```
30 PRINT AT 16,10;"■"
40 PRINT AT 17,10;"▬▬▬"
```

O. — Boââff !... Je veux bien !... Maintenant, dis-moi à ton tour ce que tu penses de mon avion à réaction :

```
50 PRINT AT 9,20;"▬▬-"
```

P. — Drôlement bien !... Dis-moi, à quoi correspond le signe moins que tu as placé devant ?

O. — C'est le radar, voyons !

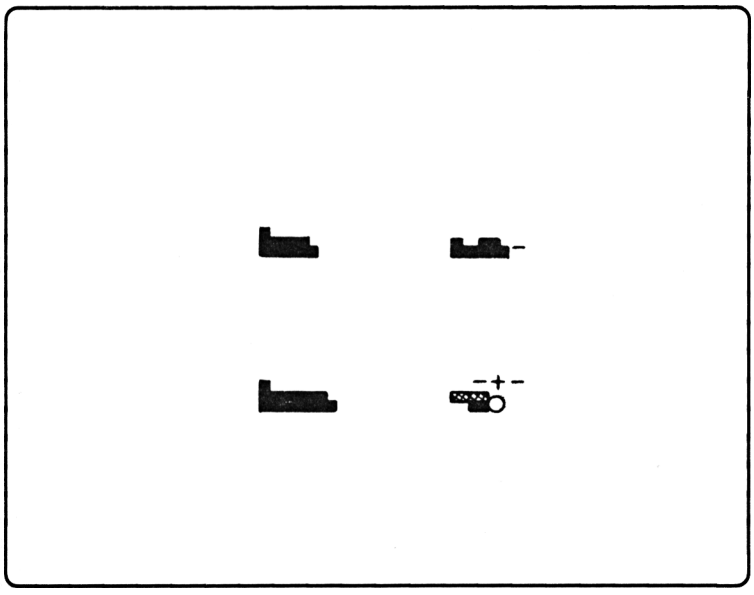
P. — Ah, bon !... Maintenant, on va essayer de terminer en beauté !... As-tu pensé à dessiner un hélicoptère ?

O. — J'y ai bien pensé à un moment, mais je n'ai rien obtenu de potable !

P. — Alors, regarde ce que je te propose :

```
60 PRINT AT 16,21;"-+-"
70 PRINT AT 17,20;"▬▬o"
```

O. — Il est super !... C'est génial le coup du rotor que tu as dessiné au-dessus !... Ça fait plus vrai que nature !



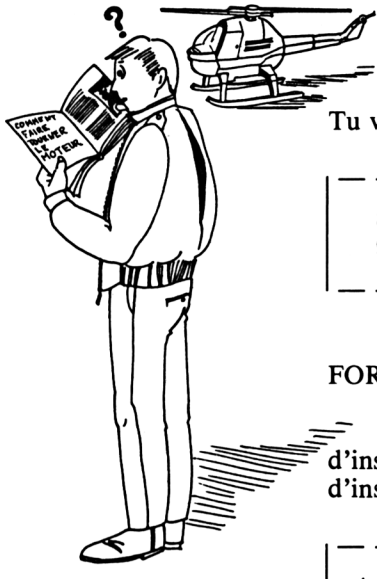
P. — A ce propos, veux-tu que je te montre ce qu'il faut faire pour obtenir l'illusion que le rotor tourne ?

O. — Oh oui !

P. — Il suffit pour cela d'ajouter une ligne d'instruction qui ne laisse apparaître que l'axe central du rotor, sans les pales, et de placer cette ligne d'instruction au milieu du programme du jeu dans lequel tu emploies l'hélicoptère.

O. — Alors, ce n'est pas faisable, puisque je n'ai pas de programme !

P. — Si !... Simplement il va falloir simuler l'existence d'un programme par deux boucles d'instructions qui ne servent à rien.



O. — Des boucles qui ne servent à rien ?... Qu'est-ce que tu veux dire par là ?

P. — Plutôt que de continuer à t'expliquer, je vais t'en écrire une tout de suite !... Tu verras, cela sera plus clair !

```
80 FOR N=0 TO 5
90 NEXT N
```

O. — Je viens effectivement de comprendre !... Tu n'as pas placé d'instruction entre FOR...TO et NEXT, ce qui fait que la boucle ne sert à rien, si ce n'est à perdre du temps.

P. — Tu as tout compris !... Telle que je l'ai écrite, elle permet de remplacer 6 lignes d'instructions !... On n'a cependant pas fini !... Après cette boucle, il faut écrire la ligne d'instruction qui ne laisse apparaître que l'axe du rotor :

```
100 PRINT AT 16,21;" + "
```

O. — Je remarque les deux espaces blancs situés de part et d'autre du signe plus. Je suppose que c'est pour effacer les deux signes moins qui représentent les pales.

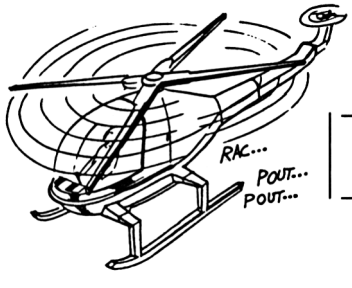
P. — Exact !... Ainsi, chaque fois que tu les imprimes et que tu les effaces, tu donnes l'impression qu'elles tournent !

O. — Alors, si je comprends bien, il faut revenir à la ligne d'instruction 60 pour boucler la boucle ?

P. — Oui, mais non sans avoir intercalé une seconde petite boucle qui ne sert à rien, sinon ton rotor aurait des ratés !

O. — Ok !... Voici donc la seconde boucle :

```
110 FOR N=0 TO 5
120 NEXT N
```



P. — Très bien !... Et maintenant voici de quoi remonter à la ligne 60 :

```
130 GOTO 60
```

O. — Je vois que tu viens de faire appel à une nouvelle instruction GOTO. En voilà au moins une qui n'est pas difficile à comprendre !

P. — Non, n'est-ce pas !... Elle te donne l'ordre d'effectuer un branchement inconditionnel vers la ligne 60 !... D'ailleurs GO TO en anglais, cela veut dire « aller à » !...

Mais, trève de discussions, fais tourner un peu ce programme qu'on voit ce que cela donne !

O. — C'est super génial !... On dirait vraiment que le rotor tourne !... Bravo !

Un train, une voiture de course et un camion

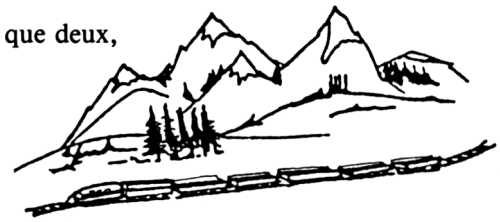
P. — Bien !... Et si maintenant on redescendait un peu sur terre !

O. — Très bonne idée !... J'ai justement étudié un train dont j'aimerais que tu me donnes des nouvelles !... Voici déjà la locomotive pour commencer :

```
10 PRINT AT 5,10;"<  >"
20 PRINT AT 6,10;"▚  ▞"
30 PRINT AT 7,10;"O  O"
```

P. — Très astucieux le fait d'employer les guillemets en vidéo inversée pour dessiner les vitres de la cabine de la locomotive !

O. — N'est-ce pas ?... Maintenant, voici les wagons !... Je n'en dessine que deux, mais tu peux en mettre autant que tu veux !



```
40 PRINT AT 6,14;"=▚▚▚▚▚=▚▚▚▚▚"
50 PRINT AT 7,14;" O  O O  O"
```

P. — Parfait !... Il est vraiment très chouette ton train !... Il ne lui manque plus qu'un rail pour rouler dessus !... Tu le dessines, ou c'est moi qui le fais ?

O. — Laisse-moi le faire !... Juste une demi-ligne horizontale à tracer et le tour est joué !

```
60 FOR C=0 TO 31
70 PRINT AT 8,C;"▬"
80 NEXT C
```

P. — Magnifique !... Je vais toutefois le proposer une légère amélioration concernant le tracé de la voie ferrée !... Remplace donc ta ligne 70 par celle-ci :

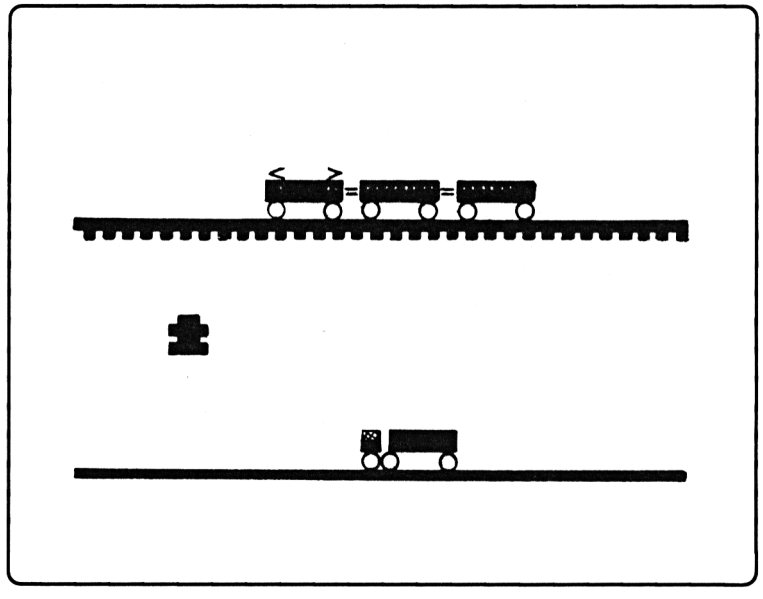
```
70 PRINT AT 8,C;"▙"
```

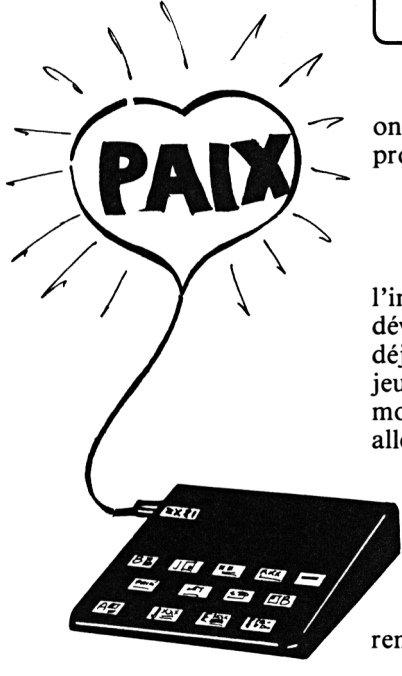
O. — Tu as raison, la voie ferrée est nettement mieux maintenant que l'on voit les traverses !... C'est comme dans la réalité !

P. — A mon tour de te montrer la petite voiture de course que j'ai inventée tout à l'heure :

```
 90 PRINT AT 13,5;"▄▄"
100 PRINT AT 14,5;"▄▄"
```

O. — Mouais !.. Ça peut effectivement passer pour une voiture de formule un !... Vivement que tu m'apprennes à faire bouger tout ça. C'est un peu statique pour le moment !





P. — Je suis d'accord avec toi, mais une chose à la fois !... Dans l'instant présent, on s'occupe de dessiner des engins !... Pour le déplacement, il faudra attendre samedi prochain !... En attendant, c'est à toi de proposer autre chose !

O. — Bien !... Alors voici un tank que je destine à des jeux de guerre !

P. — Alors là, je t'arrête tout de suite !... Les jeux vidéo sont faits pour exciter l'imagination et non pour imiter les errements de ceux qui font la guerre. Pas question de développer ensemble des thèmes basés sur la destruction, même et surtout si d'autres l'on déjà fait. Sers-toi au contraire de ce nouveau vecteur qu'est l'informatique pour créer des jeux de paix. Ils éviteront ainsi à de nouvelles pensées négatives de venir alimenter le monde fou dans lequel nous vivons !... Si, malgré ce que je viens de te dire, tu souhaites aller dans cette voie, alors tu devras le faire tout seul.

O. — Bon, bon, je n'insiste pas !... Et un camion à la place, ça peut aller ?

P. — Bien sûr !

O. — Alors, voici son programme !... Tu remarqueras. qu'il s'agit d'un semi-remorque !

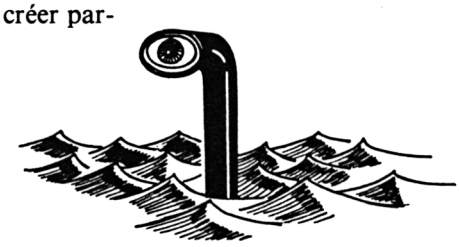
```
110 PRINT AT 19,15;"▒ ████"
120 PRINT AT 20,15;"OO   O"
```

P. — Alors là, chapeau !... Il est superbe ce camion !... Il ne lui manque plus qu'une route pour rouler dessus !

O. — Si c'est tout ce qu'il faut pour te faire plaisir !... Je vais même te la dessiner avec l'instruction PLOT, ce coup-ci !

```
130 FOR I=0 TO 63
140 PLOT I,1
150 NEXT I
```

P. — Très beau !... J'en arrive à m'étonner de ce que l'imagination peut créer parfois !... Bon, trève de bavardages, si on passait à la suite ?



O. — Et la suite, c'est quoi ?

P. — Les bateaux, mon cher, les bateaux !

Un sous-marin et des bateaux

O. — Alors, c'est moi qui débute par le programme d'un petit sous-marin :

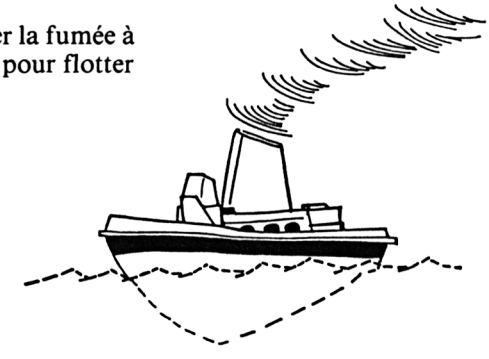
```
10 PRINT AT 12,14;"▬▟▙"
```

P. — Pas mal !... Moi, de mon côté, j'ai pensé à un navire de surface qui pourrait servir indifféremment de navire de pêche, de croisière, marchand ou océanographique.

O. — Fais voir ce navire à tout faire !

```
20 PRINT AT 0,8;"-▗▙"
30 PRINT AT 1,8;"▬▬▬"
```

O. — ... Oui, ça ressemble bien à un bateau !... Pas mal l'idée de rajouter la fumée à l'arrière de la cheminée. Ça fait plus vrai !... Il ne lui manque plus que la mer pour flotter dessus !



P. — Si ce n'est que cela, la voici :

```
40 FOR C=0 TO 31
50 PRINT AT 1,C;"-"
60 NEXT C
```

O. — Oui, c'est bien, seulement le dessin de la mer efface celui de la coque du bateau quand il passe en dessous !

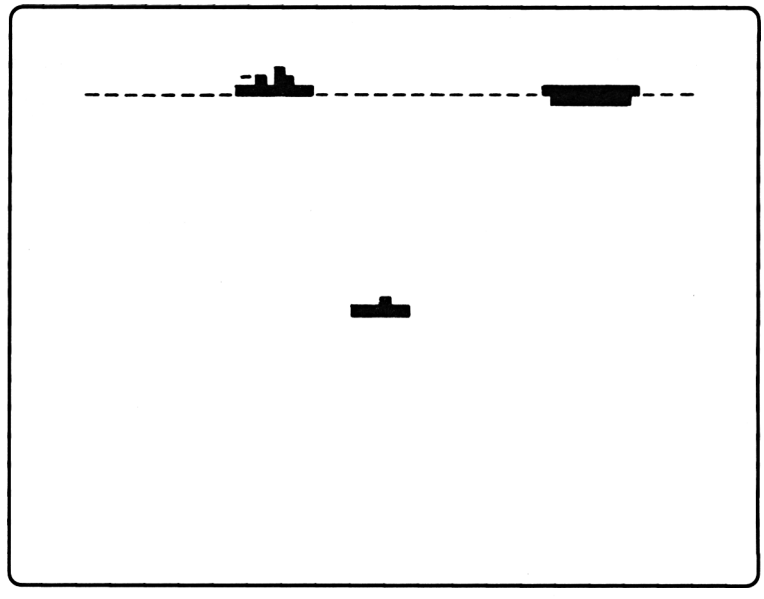
P. — Erreur de ma part !... Il est normal que cela se produise, car j'ai fait l'inverse de ce qu'il fallait faire, c'est-à-dire qu'il faut d'abord dessiner la mer et ensuite le bateau. Il ne me reste plus qu'à renuméroter les lignes 20 à 60, comme cela :

```
20 FOR C=0 TO 31
30 PRINT AT 1,C;"-"
40 NEXT C
50 PRINT AT 0,8;"-▗▙"
60 PRINT AT 1,8;"▬▬▬"
```

O. — Là, c'est nettement mieux !... A mon tour maintenant avec un porte-avions !... Je vais t'avouer que c'est la vue de ton bateau qui m'a fait pensé à ça !

```
70 PRINT AT 1,24;"▬▬▬"
```

P. — Très chouette ce porte-avions !... Il me semble être juste assez long pour pouvoir recevoir l'avion à réaction qu'on a dessiné tout à l'heure !

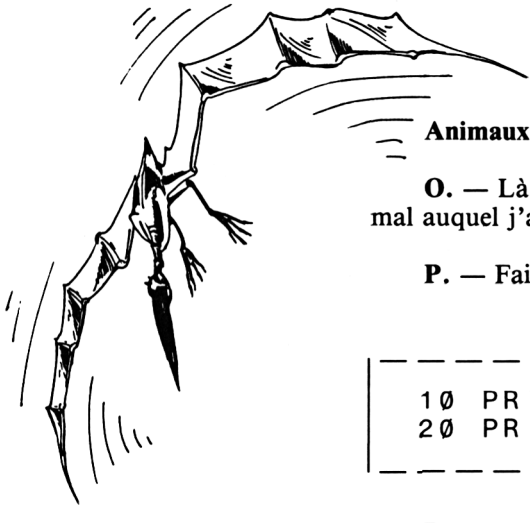


O. — C'est normal, je l'ai fait exprès !... Tu remarqueras également que j'ai supprimé tout ce qui pouvait dépasser de la piste d'appontage, de façon que les avions puissent à la fois atterrir et décoller.

P. — Bien !... As-tu encore d'autres idées d'engins à dessiner ?

O. — Non !... En tout cas, plus pour le moment !... Plus tard peut-être quand je dessinerai un jeu !

P. — Alors, il ne nous reste plus qu'à passer aux animaux et aux personnages !



Animaux et personnages

O. — Là, je dois t'avouer que je manque très nettement d'inspiration. Le seul animal auquel j'ai pensé est en fait un animal volant !

P. — Fais-le voir !

```
10 PRINT AT 6,24;"▬▬ "
20 PRINT AT 7,24;"▟▙"
```

P. — ... Pas mal !... Et je suppose que tu veux le faire battre des ailes !

O. — Comme tu me devines bien !... Est-ce que je peux reprendre la même méthode que celle qui nous a servi à faire tourner le rotor de l'hélicoptère ?

P. — Oui, elle convient parfaitement !

O. — Alors, voici le programme de battement des ailes du monstre :

```
30 FOR A=0 TO 15
40 NEXT A
50 PRINT AT 6,24;"▚▞"
60 FOR A=0 TO 15
70 NEXT A
80 GOTO 10
```

P. — Bravo !

O. — Hyper génial tu veux dire !... Et toi, tu as pensé à quoi ?

P. — Difficile de passer après ça !... Tout ce que je peux te proposer est un vulgaire cheval de course, dont le programme est le suivant :

```
 80
 90 PRINT AT 0,0;"▪   O"
100 PRINT AT 1,0;" ███"
110 PRINT AT 2,0;" █ █"
```

O. — Mouais !... Ça y ressemble plus ou moins !... Tiens, ton cheval me fait penser au départ d'une course de chevaux !

P. — Continue !

O. — Il y a une question que je me pose !... Si je veux placer cinq chevaux sur une ligne de départ, faut-il que j'écrire cinq séries de trois lignes d'instructions, une par cheval ?

P. — Très bonne question !... La réponse est non !... Il suffit de faire appel à une petite boucle bien connue !

O. — Encore !... Alors là, je ne vois pas comment !

P. — Attends, je te montre :

```
 80 FOR N=1 TO 5
 90 PRINT AT 4*N-4,0;"▪   O"
100 PRINT AT 4*N-3,0;" ███"
110 PRINT AT 4*N-2,0;" █ █"
120 NEXT N
```

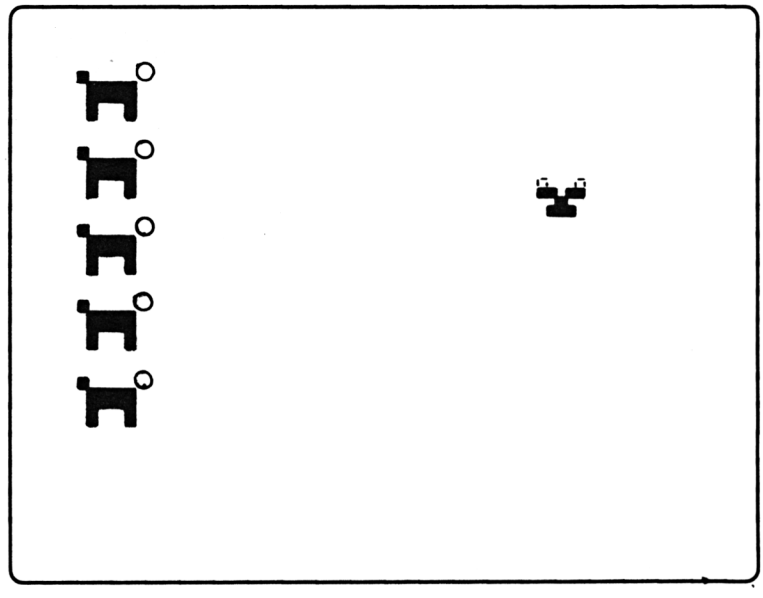
O. — Ok, j'ai compris !... Tu as placé les trois lignes d'écriture du cheval dans une boucle à cinq valeurs, et tu as remplacé les coordonnées de lignes qui étaient fixes, par la variable de boucle !... Astucieux, ça !

P. — L'écriture 4*N - 4 correspond à la ligne 0 quand N vaut 1, à la ligne 4 quand N vaut 2, et ainsi de suite de quatre en quatre !... Puisque tu as compris, je vais te demander de dessiner le numéro du dossard sur chaque cheval !

O. — Hou là là !... Qu'est-ce que tu me demandes là !... Attends que je réfléchisse !... Ça y est, j'ai trouvé !... C'est tout simple !... Il suffit d'écrire :

```
115 PRINT AT 4*N-3,1;N
```

P. — Bravo !... Je vois que tu as bonne mémoire !... Reconnais tout de même que le résultat n'est pas mal à l'écran !



O. — Oui, bien sûr...

Un parcours de golf

C'est sur cette réplique mi-figue mi-raisin que nous descendîmes manger car, après la nourriture de l'esprit, il était l'heure de penser à celle du corps. Pour une fois, le repas se déroula sans qu'il soit question d'informatique, chose assez rare il faut le dire ces temps derniers. J'en compris la raison profonde lorsque, à la fin du repas, Olivier me déclara :

O. — Comme tu m'avais dit qu'après les engins, on allait s'occuper des décors, je me suis creusé la tête pour savoir sur quoi nous allions bien pouvoir travailler cet après-midi !

P. — Et qu'en as-tu extrait ?

O. — Quelques petites idées !... J'aimerais dessiner une base lunaire... et une route... et une piste de ski... et une ceinture d'astéroïdes... et...

P. — Arrête !... Pitié pour moi !... Rien qu'avec ce que tu viens d'énumérer, j'en ai pour une semaine à moi tout seul !

O. — Mais on est deux !... C'est ça qui fait toute la différence !... Si on commençait par un parcours de golf ?

P. — Pourquoi pas !... Tu le veux comment ton « green » ?

O. — Plat, avec un trou dedans !

P. — Si c'est tout ce que tu veux, tu n'as pas besoin de moi pour cela !

O. — Ça, je le sais !... Ce que j'aimerais, c'est de pouvoir placer le trou au hasard !

P. — Là, tu commences à m'intéresser sérieusement !... Dessine donc déjà ta pelouse !

O. — C'est comme si c'était fait :

```
10 FOR C=0 TO 31
20 PRINT AT 15,C;"▩"
30 NEXT C
```

P. — Tu n'y mets pas un « tee » pour la balle ?

O. — Si, bien sûr !... Le voilà ton « tee » !

```
40 PRINT AT 15,0;"▩"
```

P. — Bien !... Maintenant, pour le trou que tu veux placer au hasard, il faut que je te parle des instructions RND et INT !

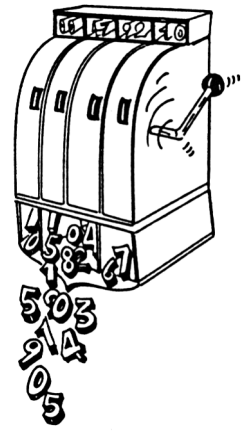
O. — Vas-y, je t'écoute !

P. — L'instruction RND est une instruction qui fabrique au hasard un nombre compris entre 0,00000000 et 0,99999999.

O. — Vraiment au hasard ?

P. — Non, pas tout à fait !... En réalité, cette instruction explore toutes les valeurs possibles, sans en oublier une seule, entre les deux valeurs limites que je t'ai indiquées et ce, dans le plus beau désordre imaginable !... Pour t'en convaincre, tape donc le programme suivant et dis-moi ce que tu obtiens :

```
1 PRINT RND
2 GOTO 1
```



O. — J'obtiens toute une série de valeurs compliquées !

P. — As-tu l'impression qu'elles sortent au hasard ?

O. — Tout à fait !... Il n'y en a pas deux qui se suivent !

P. — Bien !... Il te faut maintenant exploiter ces valeurs, car tel quel, tu ne peux rien en faire !

O. — C'est bien ce qu'il me semblait !

P. — Suppose donc que tu veuilles obtenir un numéro au hasard entre 0 et 9, grâce à cette instruction RND. La première opération que tu dois effectuer consiste tout d'abord à multiplier RND par le nombre de valeurs différentes que tu veux obtenir.

O. — Dans ce cas, c'est 10 !... Il faut donc que j'écrive :

```
1 PRINT RND*10
2 GOTO 1
```

P. — C'est ça !... Tu peux maintenant observer à l'écran que tu obtiens toute une série de nombres compris entre Ø,ØØØØØØØØ et 9,9999999.

O. — C'est le cas !

P. — Bien !... Alors, il ne te reste plus qu'à te débarrasser de la partie décimale de chaque nombre en employant l'instruction INT qui veut dire « partie entière ».

O. — Au fait, que voulait dire RND ?

P. — Les trois lettres RND proviennent de la contraction du mot anglais « random » qui signifie « au hasard » !

O. — Dans le fond, le BASIC, c'est tout simple quand on connaît l'anglais...

P. — Que je te dise deux mots sur la syntaxe de cette instruction INT. Il faut mettre entre parenthèses le nombre décimal dont tu veux la partie entière !

O. — Comme ça ?

```
 1 PRINT INT (RND*1Ø)
 2 GOTO 1
```

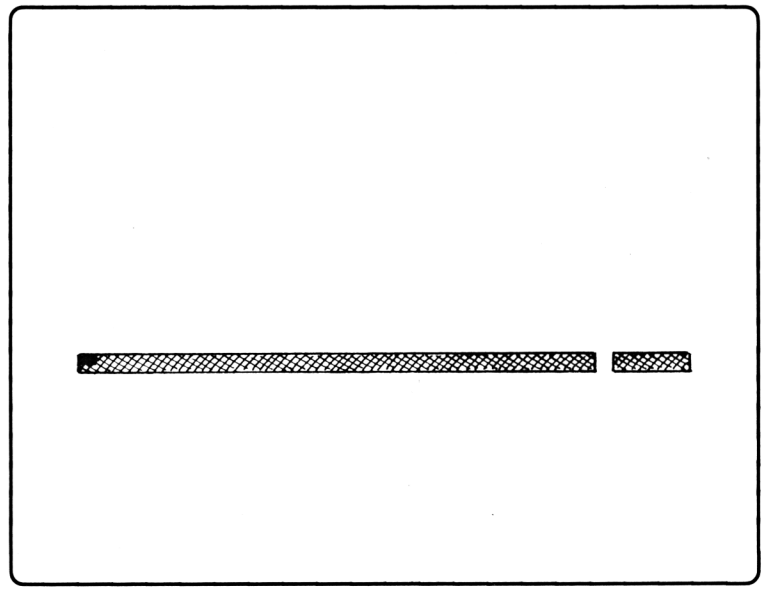
P. — Oui, très bien !... Comme tu peux le voir, tu obtiens tous les chiffres entre Ø et 9 dans le désordre le plus parfait !

O. — J'aurais jamais cru qu'on pouvait fabriquer « le hasard » aussi facilement !

P. — N'est-ce pas ?... Et si maintenant tu terminais ton parcours du golf !... Tu disposes de tout ce dont tu as besoin pour cela !

O. — Ok !... J'efface les lignes 1 et 2 qui ne me servent plus à rien et j'écris la ligne de dessin du trou :

```
 5Ø PRINT AT 15,INT (RND*1Ø+2Ø)
;"  "
```



P. — Je vois que tu places le trou au hasard entre la 2Ø et la 29e colonne !... Cela te fait dix emplacements possibles !

O. — Eh oui !... Comme cela, ça rendra la partie plus difficile à jouer, puisque le trou ne sera jamais placé au même endroit !

P. — Filou !... Un petit conseil toutefois !... Il vaut mieux poser une variable qui définit la position du trou. Cela facilitera plus tard le repérage de la position de la balle par rapport au trou !

```
50 LET T=INT (RND*10+20)
60 PRINT AT 15,T;" "
```

O. — Bien, je note !... Maintenant, si tu n'as rien d'autre à me dire au sujet du parcours de golf, on peut peut-être passer à la caverne !

La caverne

P. — Allons-y pour la caverne !... Explique-moi un peu comment tu la veux !

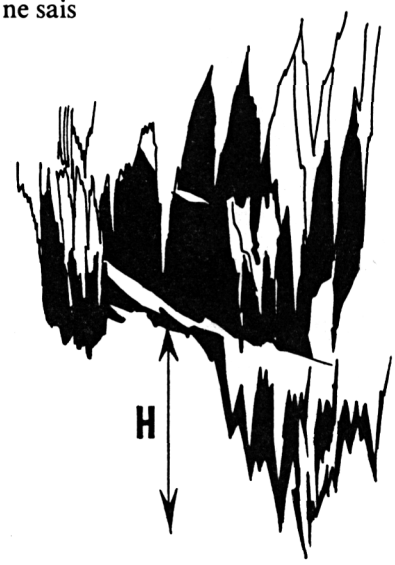
O. — Je la voudrais avec un plafond tout plein de stalactites dont la hauteur au-dessus du sol serait toujours différente !

P. — Quel programme tu me proposes là !... En tout cas, je vois avec plaisir que depuis que tu sais maîtriser l'instruction RND, tu n'hésites plus à la mettre à toutes les sauces !... Pour commencer, si tu veux bien, on ne va dessiner que le plafond de la caverne !... Le reste, on l'écrira au fur et à mesure !

O. — D'accord !... Alors c'est toi qui commences par le plafond, car moi, je ne sais pas le faire !

P. — Bien sûr !... Voici donc un plafond mobile :

```
10 LET H=6
20 FOR C=0 TO 31
30 LET A=INT (RND*5)-2
40 LET H=H+A
50 FOR L=0 TO H
60 PRINT AT L,C;"■"
70 NEXT L
80 NEXT C
```



O. — Hou là là, que c'est compliqué !

P. — Ne t'inquiète pas !... Je vais t'expliquer cela en détail !... Tout d'abord, j'ai fixé la hauteur du plafond par la variable H.

O. — Je suppose que la valeur H = 6 correspond à la ligne sur laquelle tu commences à dessiner le plafond.

P. — Exact !... Maintenant, il faut que je fasse varier la hauteur du plafond. J'obtiens cela à l'aide de la ligne 3Ø.

O. — Attends, il y a plusieurs choses que je ne comprends pas !... D'abord, tu n'as pas placé les parenthèses de l'instruction INT de la même façon que tout à l'heure !

P. — C'est exprès !... J'ai fait cela pour te montrer qu'il y a deux écritures possibles !... Dans le cas présent, la valeur de A peut varier entre -2 et +2.

O. — Pourquoi avoir choisi ces valeurs ?

P. — Pour obtenir des stalactites suffisamment longues, comme tu me l'as demandé !... Si A n'avait varié qu'entre −1 et +1, leur longueur aurait été trop faible !

O. — Ah bon !... Et pourquoi avoir placé cette ligne d'instruction à l'intérieur de la grande boucle et non à l'extérieur ?

P. — Pour que la valeur de A change à chaque colonne !... Si j'avais placé cette ligne avant la ligne 2Ø, j'aurais obtenu un plafond entièrement horizontal !

O. — C'est vrai ça !... Tout compte fait, il n'est pas compliqué ton programme !... Je vais d'ailleurs l'essayer !

Et Olivier s'amusa à dessiner toute une série de plafonds dont le relief était différent à chaque fois. Après plusieurs essais, il me demanda :

O. — Par moments, il se passe un phénomène bizarre !... Lorsque le plafond monte vers le haut de l'écran, il disparaît de temps en temps !

P. — Ça, c'est normal !... Cela se produit lorsque l'opération de la ligne 4Ø donne pour H un résultat inférieur à Ø !... Et comme l'ordinateur ne sait pas imprimer en dehors de l'écran...

O. — Comment peut-on faire pour remédier à cela ?

P. — En posant une butée telle que celle-ci :

```
45 IF H<1 THEN LET H=1
```

O. — Effectivement, ça marche !... A la place des trous dans le plafond, j'obtiens maintenant des surfaces planes.

P. — ... qui ne sont pas très réalistes, il faut le dire !... Dans ces conditions, je sais t'enseigner une astuce qui va les rendre moins planes en les remplissant de stalactites !... Il te suffit de remplacer H = 1 par H = H+2.

O. — Comme ça ?

```
45 IF H<1 THEN LET H=H+2
```

P. — Exactement !... Alors, qu'en penses-tu ?

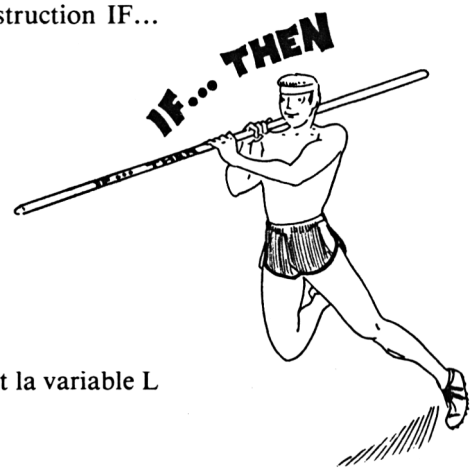
O. — Très astucieux !... On ne se rend plus compte de rien !

P. — Je suis heureux que cela te plaise !... Pour continuer avec le sol de la caverne, je te propose de maintenir une hauteur constante entre le sol et le plafond !... Cela te convient-il ?

O. — Bien sûr !... Il vaut mieux ne pas trop compliquer au début !... Tu me montres comment il faut faire !

P. — Il te faut employer la même méthode que celle qui t'avais permis de tracer les buts de ton terrain de football !

O. — Tu veux dire qu'il faut que j'effectue un saut à l'aide de l'instruction IF... THEN ?



P. — Exactement !... Tu penses pouvoir l'écrire tout seul ?

O. — Oui !... Comme ça, c'est correct ?

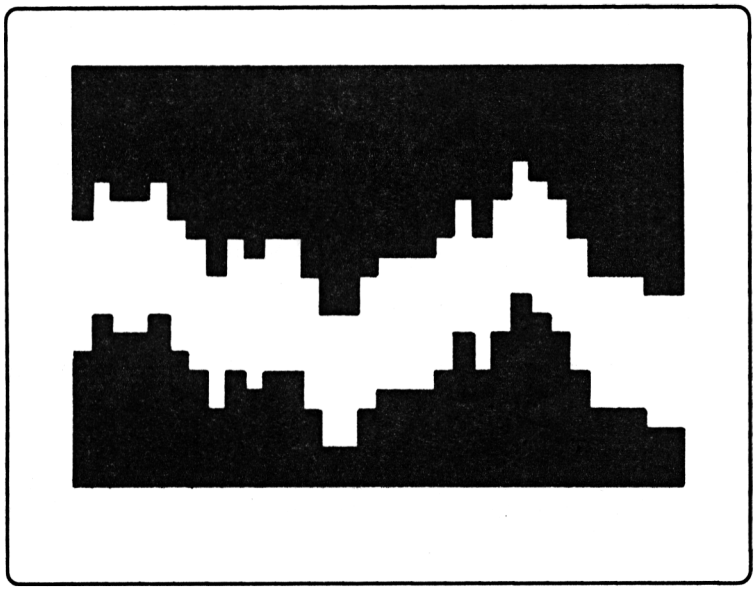
```
55 IF L=H THEN LET L=H+7
```

P. — Tout à fait !... N'oublie pas de modifier la ligne 5Ø en autorisant la variable L à aller jusqu'à 21.

O. — Voilà, c'est fait !

```
5Ø FOR L=Ø TO 21
```

P. — Je constate au passage que tu as décidé de maintenir une hauteur constante de 7 carreaux !... Es-tu content du résultat ?



O. — Oui !... C'est pas mal !... Par contre, j'ai à nouveau les mêmes problèmes dans le bas de l'écran que j'avais tout à l'heure dans le haut.

P. — Alors, même motif, même punition !... Il te faut ajouter une seconde butée, mais cette fois-ci vers le bas !

O. — Un truc dans le genre :

```
47 IF H>21 THEN LET H=21
```

P. — Oui !... Mais tu vas remarquer qu'une telle butée n'empêche pas la caverne de se rétrécir au voisinage du sol. Pour maintenir l'écart constant, il te faut l'écrire ainsi :

```
47 IF H>14 THEN LET H=14
```

P. — ... ou encore, si tu n'aimes pas les sols plats :

```
47 IF H>14 THEN LET H=H-2
```

O. — Ok, j'ai compris !... Tu as choisi la valeur 14, car elle correspond à 21 – 7, c'est-à-dire à la dernière ligne du bas de l'écran moins la hauteur de la caverne !

P. — Exactement !... Pour toute autre hauteur, il suffit d'effectuer la soustraction !

O. — Vivement que je puisse faire voler un animal préhistorique dans la caverne qu'on vient de construire !

P. — Patience, ça vient !... Mais avant d'en arriver là, finissons d'abord les décors !... Quel est le suivant au programme ?

O. — Une ville !...

Une ville

O. — ... je voudrais dessiner une ville avec plein d'immeubles !

P. — C'est pas follement gai comme environnement !... Tu n'en veux pas plutôt une avec des pavillons ?

O. — Pourquoi pas !... Mais après les immeubles seulement !

P. — Bon !... Eh bien, puisque c'est ton souhait, exauçons-le !... Que désires-tu comme hauteur pour les immeubles ?

O. — Je ne sais pas, moi !... De 1 à 6 carreaux, ça devrait aller !

P. — Alors, ce que nous allons faire, c'est de dessiner les immeubles à partir d'une cellule de base comportant une fenêtre, comme celle-ci :

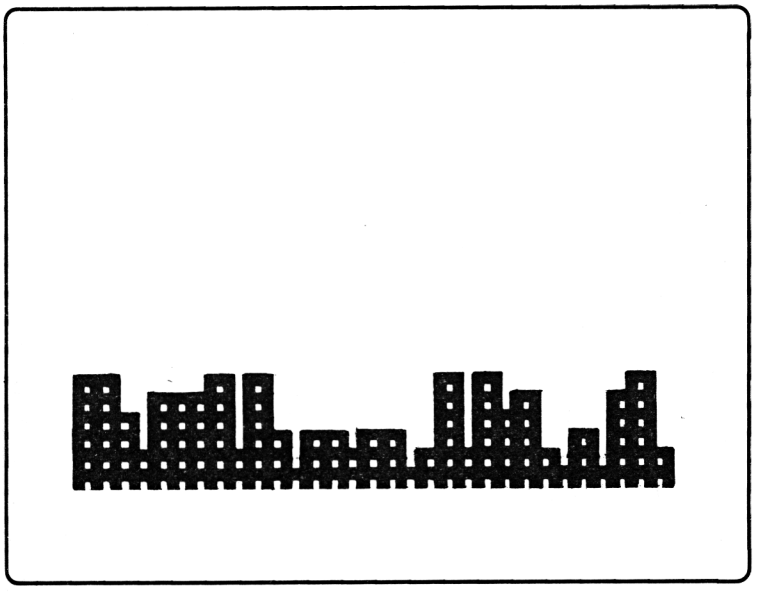
```
"▟"
```

O. — Ça me paraît très bien !... Ainsi, en les empilant les unes sur les autres, on obtiendra une ville du plus bel effet !... Tu me montres comment il faut faire !

P. — Voici donc le programme attendu :

```
10 FOR C=0 TO 30
20 LET A=INT (RND*6)+16
30 FOR L=A TO 21
40 PRINT AT L,C;"▟ "
50 NEXT L
60 NEXT C
```

P. — ... Tu noteras que je ne dessine que jusqu'à la 30e colonne !... Cela provient du fait que la cellule de base occupe un carreau et demi et déborde dans la colonne suivante.

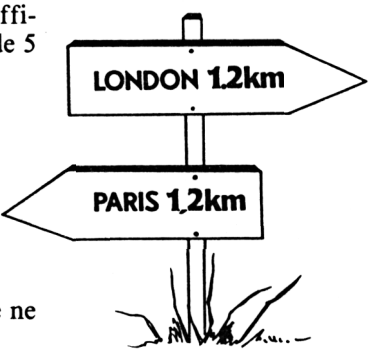


O. — Effectivement !... Dis-moi aussi pourquoi tu as ajouté le nombre 16 après le calcul de la hauteur de chaque immeuble ?

P. — Parce que les lignes sont numérotées du haut vers le bas, et non l'inverse !... Maintenant, veux-tu que je te montre un truc ?

O. — Bien sûr !

P. — Il existe un moyen pour créer des rues de façon à séparer les immeubles en pâtés !... Il suffit pour cela d'ajouter à la ligne 10 l'instruction STEP, suivie d'un coefficient de valeur comprise entre 1 et 2 !... Avec la ligne suivante, tu obtiens des blocs de 5 immeubles :




```
10 FOR C=0 TO 30 STEP 1.2
```

O. — C'est marrant comme tout, ce truc là !... Mais il y a quelque chose que je ne comprends pas !... Pourquoi écrire 1.2 et non 1,2 ?

P. — Ah, c'est vrai !... J'ai oublié de te dire que les anglo-saxons utilisent tous le point à la place de la virgule pour séparer la partie entière de la partie fractionnaire des nombres.

O. — Comme sur les calculatrices ?

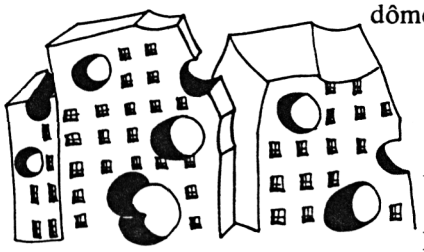
P. — Pareil !... Veux-tu l'essayer tout seul avec d'autres valeurs de coefficient ?

O. — Oui !... Et d'ailleurs, si tu permets, je vais aussi modifier la hauteur des immeubles !

P. — Fais comme tu veux !... Le programme est à toi.

Et c'est ainsi qu'Olivier s'amusa à transformer la ville en une sorte de gruyère aux contours irréguliers, rien qu'en modifiant la hauteur et l'emplacement des immeubles !... Puis, tout d'un coup, il s'écria :

O. — Je viens d'avoir une idée géniale !... Il s'agit d'une base lunaire avec un grand dôme de protection !



Une base lunaire

P. — Et tu te sens capable de faire ça tout seul !

O. — Je vais essayer !... Tout d'abord, je voudrais dessiner un demi-cercle !... Comment fait-on pour cela ?

P. — Il te faut employer pour cela les instructions SIN et COS !

O. — Hou là là !... C'est pas des fonctions trigonométriques, ça ?

P. — Si !... Et après ?... Sache que pour dessiner un cercle, il suffit de faire varier l'angle au centre entre Ø et 2π et d'écrire pour les coordonnées des lignes et des colonnes : I = R * COS (A) et J = R * SIN (A).

O. — A quoi correspond la lettre R ?

P. — Au rayon du cercle !...

O. — Et la lettre A entre parenthèses ?

P. — A la valeur de l'angle exprimée en radians !... Or dans un cercle, il existe 2 π radians, ce qui équivaut à 360 degrés !

O. — Bien !... Alors, voici un demi-cercle de rayon 12 :

```
10 FOR A=Ø TO PI
20 PLOT R*COS (A),R*SIN (A)
30 NEXT A
```

O. — ... Tiens, j'obtiens un message d'erreur 2/2Ø !

P. — Cela veut dire qu'une des variables que tu emploies à la ligne 2Ø n'a pas été définie !

O. — C'est vrai, j'ai oublié de définir R !... Voici donc l'oubli réparé :

```
5 LET R=12
```

O. — ...Seulement, ça ne marche toujours pas !... Le dessin est complètement tassé dans le coin gauche en bas de l'écran !...

P. — C'est normal, car tel que tu as écrit ton programme, tu as placé le centre du cercle sur le point de coordonnées Ø, Ø !... Il faut le décaler vers la gauche d'au moins 12 demi-carreaux, pour qu'il soit complètement visible !

O. — Voilà, c'est fait !... J'ai ajouté + 16 à la coordonnée de colonne de la ligne 2Ø :

```
20 PLOT R*COS (A)+16,R*SIN (A)
```

O. — ... Décidément, je ne serai jamais content !... Maintenant, j'ai bien un demi-cercle, mais il n'a que 4 points !

P. — Ça, c'est de ma faute !... J'ai oublié de te dire que si tu ne définis pas le pas de progression à la ligne 1Ø, il est d'office de π/3 !... C'est ce qui explique le fait que tu n'obtiennes que 4 points sur l'écran !

O. — Et de combien il doit être, le pas de progression ?

P. — A toi de le trouver par tâtonnements successifs !... Ce qu'il faut, c'est trouver une valeur suffisamment élevée pour qu'il n'y ait plus de trous dans le dessin, mais pas trop forte pour éviter que les carrés ne se soudent entre eux !... Essaye avec π/30 !

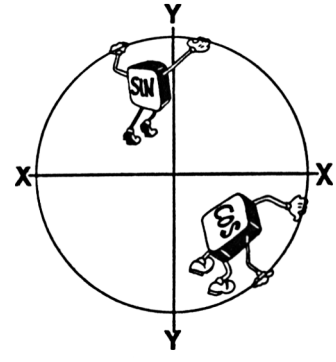
O. — C'est pas suffisant !... Il reste des trous dans le dessin !... Voilà, avec π/34, c'est bon !...

```
10 FOR A=0 TO PI STEP PI/34
```

P. — C'est pas mal !... Et maintenant, qu'est-ce que tu comptes faire ?

O. — Dessiner le même à côté et les relier par un tunnel !... Voilà déjà pour le second dôme :

```
25 PLOT R*COS (A)+48,R*SIN (A)
```



P. — Très bien !... Je vois que tu l'as décalé de 32 demi-carreaux par rapport à l'autre !... Par contre, tu peux simplifier ton écriture en posant deux variables I et J de coordonnées, ce qui t'évite d'écrire deux fois les instructions SIN et COS !

```
20 LET I=R*COS (A)
30 LET J=R*SIN (A)
40 PLOT I+16,J
50 PLOT I+48,J
60 NEXT A
```

O. — Effectivement, c'est plus élégant !... Il me reste toutefois encore un problème à résoudre. Il manque un demi-carreau à la partie gauche du demi-cercle.

P. — Ce phénomène provient de la règle d'arrondi employée par le système d'affichage !... Tu peux le masquer en dessinant le sol lunaire en dessous !

O. — Très bonne idée !... Et un sol qui marche, un !

```
70 FOR I=0 TO 63
80 PLOT I,0
90 NEXT I
```

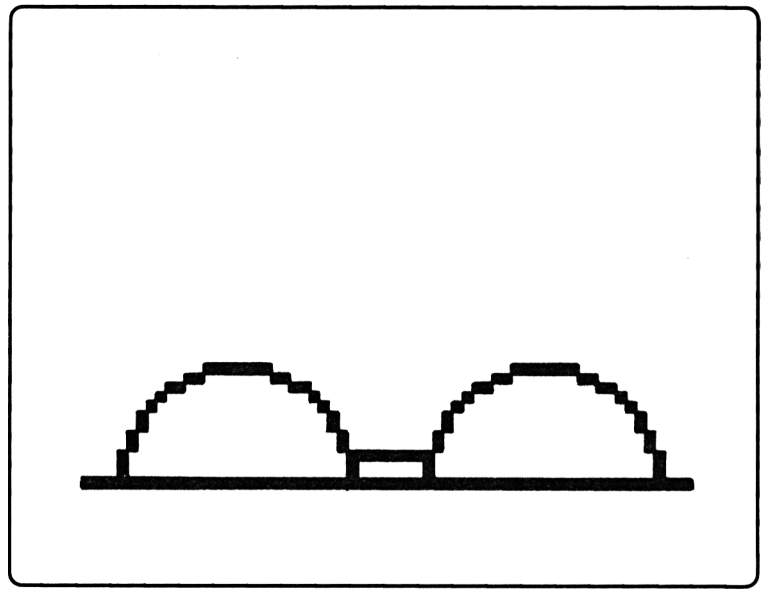
P. — Tu vois, le défaut ne se voit plus !... Il ne reste plus qu'à dessiner le tunnel et tu auras terminé !

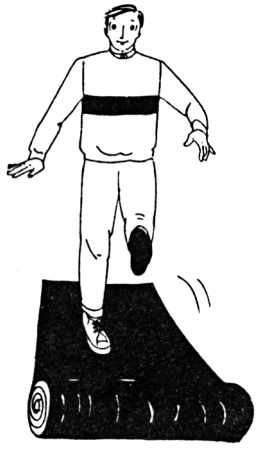
O. — C'est comme si c'était fait !

```
100 FOR I=29 TO 35
110 PLOT I,3
120 NEXT I
```

P. — Très belle ta base lunaire !... Bravo !... Et si tu essayais de faire les dômes plus petits !... De 10 ou de 8 demi-carreaux de rayon par exemple !

O. — Bonne idée !





Là-dessus, je quittai un instant Olivier, bien décidé à m'octroyer quelques minutes de détente, propres à renouveler les énergies créatrices... J'en étais à la moitié de mon verre quand une voix venant d'en haut me rappela à l'ordre :

P. — Je suis prêt pour la route !

Là, il fallait comprendre qu'il ne s'agissait pas de partir en voyage...

Une route

P. — Deux secondes, j'arrive !... J'en connais un qui va être content !... Toi qui trouvais que tout ce que nous dessinions était un peu trop inanimé, tu vas enfin créer un décor qui bouge !

O. — Pourquoi, il est possible de dessiner une route qui avance toute seule ?

P. — Oui, tout à fait !... Il suffit pour cela de faire appel à l'instruction SCROLL !

O. — Décidément, elles ont de ces noms les instructions BASIC !... Comment prononces-tu ça ?

P. — S... C... R... O... L... L..., SCROLL !... Cette instruction permet de faire remonter d'une ligne la totalité du contenu de l'écran.

O. — Et à quoi cela peut-il servir ?

P. — A libérer de la place dans le bas de l'écran pour pouvoir continuer à écrire un texte, quand tout l'écran est rempli !

O. — Ah bon !... Et que devient la ligne du haut qui disparaît de l'écran ?

P. — Elle est conservée dans la mémoire de programmation. Par contre, elle est effacée de la mémoire d'affichage !

O. — Bien !... Mais je ne vois pas le rapport que cela peut avoir avec une route qui avance toute seule !

P. — Imagine que tu ne dessines qu'une tranche de route sur la ligne du bas de l'écran et qu'ensuite, grâce à l'instruction SCROLL, tu fasses remonter cette tranche d'une ligne, tu obtiens ainsi de ligne en ligne une route qui se déplace du bas vers le haut de l'écran.

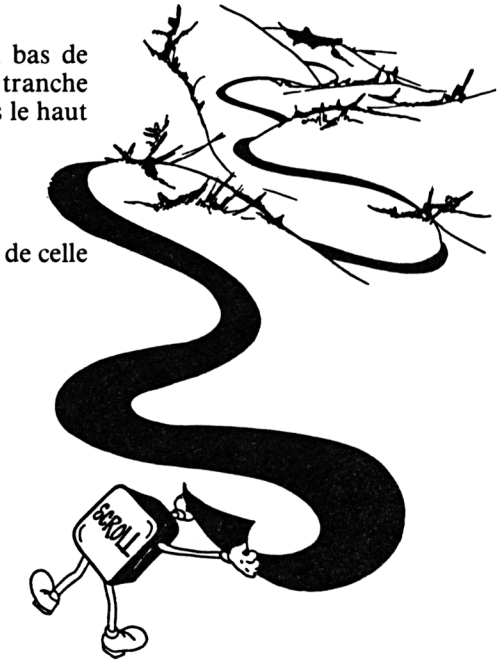
O. — Je commence à comprendre !

P. — Il te suffit alors de redessiner une nouvelle tranche de route en dessous de celle qui a grimpé d'une ligne et tu obtiens ainsi une route complète.

O. — Tu me montres ?

P. — Tout de suite !... Voilà déjà pour un début :

```
10 PRINT AT 21,10;"
"
20 SCROLL
30 GOTO 10
```



O. — C'est pas génial, ton truc !... Une fois que la route est entièrement dessinée sur l'écran, on n'a plus la moindre impression de mouvement !

P. — C'est normal, tu sais !... Cela est dû au fait que tu ne disposes d'aucun point de repère !

O. — Comme une ligne blanche interrompue par exemple ?

P. — Oui !... Ce sont justement les bandes blanches qui, en défilant, produisent l'illusion que la route entière se déplace.

O. — Et comment fait-on pour dessiner une ligne en pointillé ?

P. — En dessinant alternativement une tranche de route sans pointillé et une avec pointillé !... Comme cela par exemple :

```
25 PRINT AT 21,10;"
"
27 SCROLL
```

O. — Super !... Voilà au moins du mouvement !... Pourtant je lui ferais un tout petit reproche à ta route !

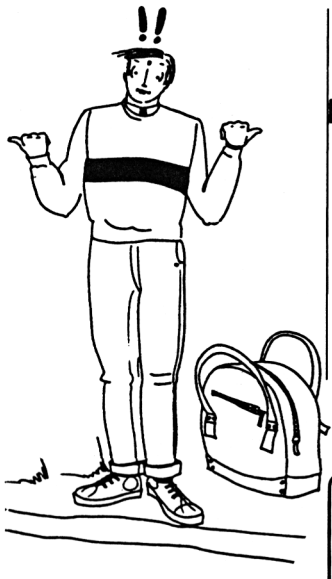
P. — Ah oui, lequel ?

O. — Le sens de défilement de la route n'apparaît pas clairement !... On ne sait pas si elle monte ou si elle descend !

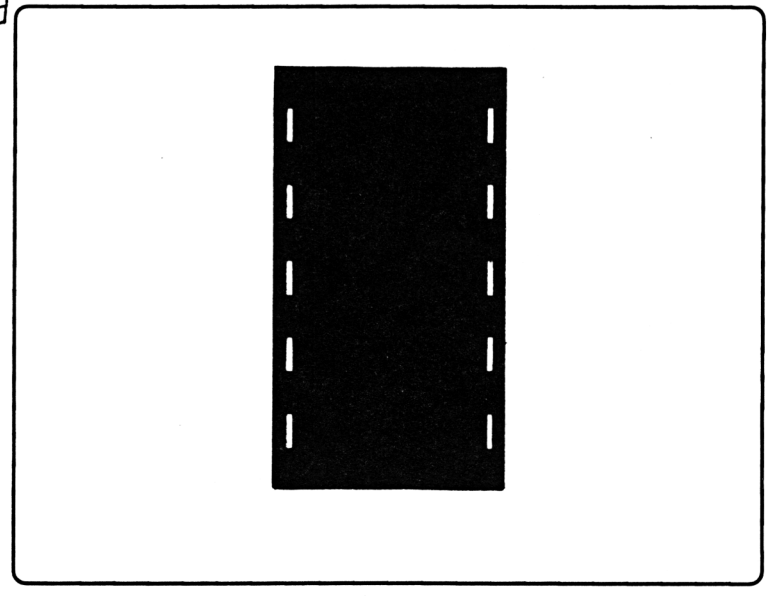
P. — Ce phénomène provient de ce que le pointillé n'est que d'un carreau de longueur. Avec deux carreaux ou plus, il n'y a plus de problème !... L'ambiguïté a automatiquement disparue !

O. — Et comment fait-on pour dessiner des pointillés de plusieurs carreaux de longueur ?

P. — A l'aide de deux boucles, une pour la partie noire et une pour la partie blanche !... Voici d'ailleurs le programme d'un pointillé de deux carreaux de longueur :



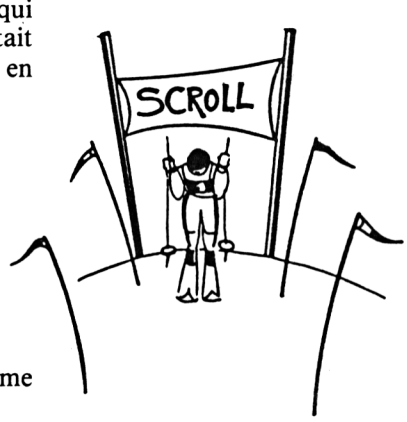
```
10 FOR A=1 TO 2
20 PRINT AT 21,10;"
"
30 SCROLL
40 NEXT A
50 FOR A=1 TO 2
60 PRINT AT 21,10;"
"
70 SCROLL
80 NEXT A
90 GOTO 10
```



O. — Hyper génial !... Je peux m'amuser à modifier ?

P. — Autant qu'il te plaira !... Comme cela, tu ne t'en souviendra que mieux !

Et Olivier partit dans une invraisemblaole série de lignes en tout genre, ponctuée de quelques commentaires visant à définir leurs ressemblances avec la réalité. Ce fut moi qui cette fois le rappelai à l'ordre. Je voyais la journée s'avancer rapidement et il restait encore diverses petites choses à terminer, si j'en croyais le programme que j'avais en main, et qui disait qu'à cette heure-ci, nous devrions être sur une piste de ski...



Une piste de ski

P. — Es-tu prêt à me suivre en slalom sur la neige ?

O. — Prêt !... Crois-tu qu'il va être possible de faire bouger la piste de ski comme nous l'avons fait pour la route ?

P. — Pareil !... On va à nouveau faire appel à l'instruction SCROLL !

O. — Alors, dans ces conditions, je devrais pouvoir le faire tout seul !... Voilà déjà un petit programme qui devrait permettre de tracer une piste de ski :

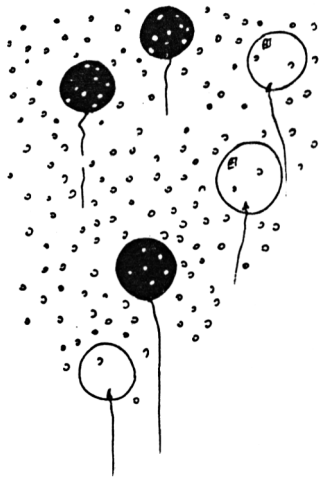
```
10 PRINT AT 21,10;"*       *"
20 SCROLL
30 GOTO 10
```

P. — Bravo !... Je vois que ta piste fait 5 carreaux de largeur !... Ce sera peut-être un peu juste si tu dessines un skieur qui doit demeurer dessus !... Quelle bonne idée aussi d'avoir choisi l'étoile pour la border ; cela fait tout de suite penser à la neige !

O. — Ça, c'était facile !... Le problème, maintenant, c'est de dessiner des virages !... Tu me souffles une idée !

P. — Ok !... Remplace donc dans la ligne 10 la coordonnée de colonne par une variable qui change de valeur au hasard !

O. — Ah, merci !... Voici donc une piste toute en virages :



```
10 PRINT AT 21,INT (RND*32);"*
"
```

P. — Tu crois ?... On dirait plutôt un grand lâcher de ballons !

O. — J'ai pourtant fait ce que tu m'as dit !

P. — Quand je t'ai parlé d'une variable qui changeait de valeur, j'imaginais que tu avais saisi qu'elle ne devait augmenter ou diminuer que d'une unité à chaque ligne, de façon à assurer une continuité dans le dessin de la piste.

O. — Et comment est-ce qu'on fait ça ?

P. — En fixant une valeur de début à ta variable et en y ajoutant + 1 ou –1 à chaque parcours de la boucle du programme.

O. — Quelque chose comme ça ?

```
10 LET X=10
20 LET A=INT (RND*3)-1
30 LET X=X+A
40 PRINT AT 21,X;"*       *"
50 SCROLL
60 GOTO 20
```

P. — Exactement !... Il faut placer, comme tu l'as fait, la valeur de départ en dehors de la boucle et celle qui doit changer tout le temps en dedans !

O. — Ça, j'avais compris !... Par contre, je ne suis pas encore satisfait du résultat !... Je trouve que la piste ne « slalomme » pas assez !

P. — Moi, aussi !... Attends, on va arranger cela en ajoutant une petite boucle dans la grande, qui va obliger la variable à conserver la même valeur de changement pendant plusieurs lignes :

```
25 FOR B=0 TO INT (RND*3)+1
55 NEXT B
```

O. — Whaooo, l'astuce !... Attends que je comprenne : B peut varier entre 0 et un chiffre qui peut être compris entre 1 et 3. Donc, au total, cela fait des virages qui durent de 2 à 4 lignes !

P. — Exact !... C'est d'ailleurs ce que tu peux constater à l'écran !

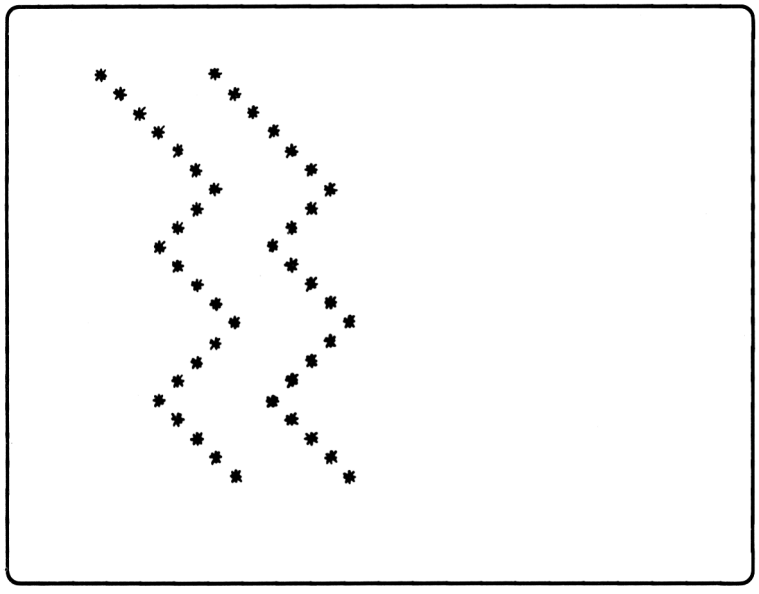
O. — Oui !... Quoique ce n'est pas encore parfait !... Il reste encore à placer des butées à droite et à gauche.

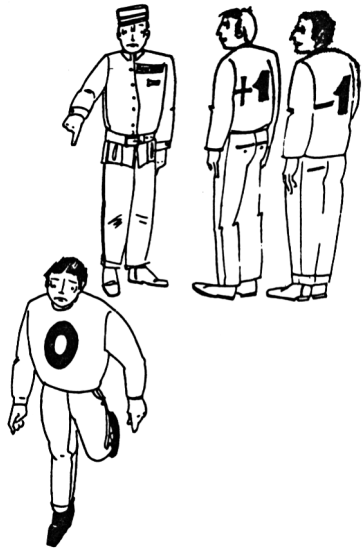
P. — Ça, tu sais le faire !... Tu n'as qu'à te souvenir de ce qu'on avait fait pour la caverne !

O. — C'est vrai !... Alors, voici les butées :

```
35 IF X<0 THEN LET X=0
37 IF X>25 THEN LET X=25
```

P. — Bien !... Est-ce qu'enfin la piste te plaît ?





O. — Presque !... Je voudrais supprimer les parties verticales droites. Est-ce que tu penses que c'est possible ?

P. — Tout à fait !... Elles apparaissent chaque fois que A est égal à Ø. Or il existe une méthode toute simple pour empêcher A de valoir Ø !

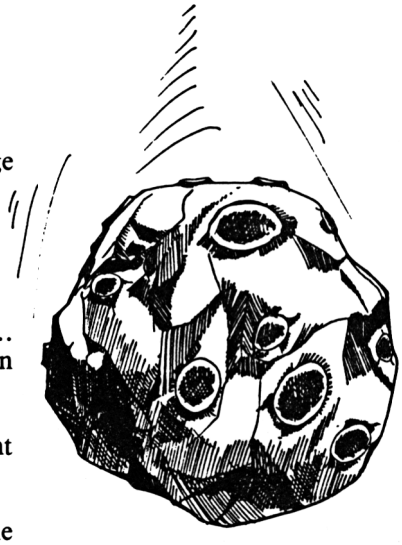
O. — Laquelle ?

P. — Elle consiste à refuser la valeur de A chaque fois qu'elle vaut Ø, grâce à une ligne du genre :

```
22 IF A=Ø THEN GOTO 2Ø
```

O. — Je comprends !... C'est vraiment plein d'astuces la programmation !

P. — Pas plus qu'autre chose, tu sais !... Ce n'est qu'une question d'apprentissage de connaissances !... Et maintenant, que faisons-nous ?



Une ceinture d'astéroïdes

O. — Le lâcher involontaire de ballons de tout à l'heure m'a donné une idée !... J'aimerais dessiner une ceinture d'astéroïdes qu'il faudrait que je franchisse avec mon vaisseau spatial.

P. — Eh bien, cela me paraît être tout à fait réalisable !... Je dirais même que l'ayant déjà fait une fois, tu dois pouvoir programmer cela tout seul !

O. — C'est bien ce que je comptais faire !... Je vais tout simplement réécrire le même programme, et s'il présente des défauts, je l'améliorerai pas la suite !

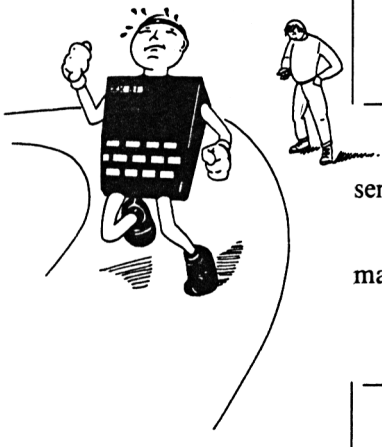
```
1Ø PRINT AT 21,INT (RND*32);"*
"
2Ø SCROLL
3Ø GOTO 1Ø
```

P. — Alors, comment tu la trouves ta ceinture d'astéroïdes ?

O. — Pas assez dense !... Ça va être trop facile de passer à travers !...

P. — Si ce n'est que cela qui te chagrine, je te suggère de placer ton instruction PRINT dans une petite boucle pour augmenter la densité d'astéroïdes !... Quelque chose comme :

```
 5 FOR A=1 TO 1Ø
15 NEXT A
3Ø GOTO 5
```



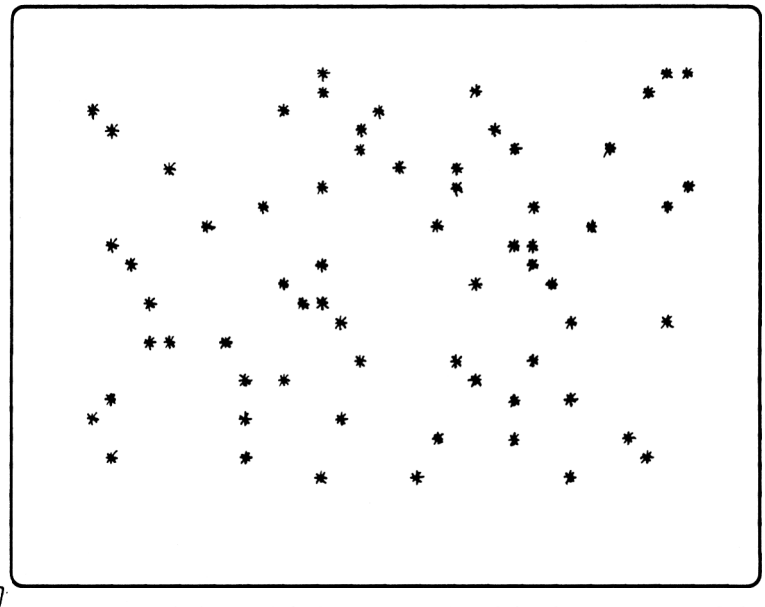
O. — Oui, mais maintenant, c'est trop dense !... Je n'ai même plus la place de passer entre les astéroïdes !

P. — Alors, réduis le nombre d'astéroïdes par ligne, en diminuant la valeur maximale de A !... Je te laisse découvrir la valeur qui convient le mieux !

O. — Eh bien, cela me semble être A = 3 !

```
 5 FOR A=1 TO 3
```

P. — As-tu remarqué que la vitesse de défilement des astéroïdes dépendait de leur densité ?



O. — Oui, je voulais justement t'en faire la remarque. Cela provient de quoi ?

P. — Du temps que met l'ordinateur à parcourir la boucle !... Plus il doit effectuer de tours, comme pour A = 1 TO 1Ø, plus cela prend de temps et réduit donc la vitesse de défilement.

O. — Tout ça me donne envie de dessiner une soucoupe en train d'essayer de se frayer un chemin. On a le temps, tu crois ?

P. — Non, pas aujourd'hui !... Il se fait tard !... Je te propose de faire cela samedi prochain !... On en profitera d'ailleurs pour étudier en détail la façon de créer le mouvement !

O. — Alors, vivement samedi prochain !

ANIMONS LES IMAGES

Aux dires d'Olivier, cette semaine d'attente fut l'une des plus longues qu'il eut à vivre. Pourtant, tout arrive et c'est tout juste si, tiré du lit au petit matin d'un samedi tout neuf, j'eus le droit de prendre mon petit déjeuner dans des conditions acceptables, tellement il était pressé que je m'asseye devant un écran d'ordinateur plutôt que devant un bol de chocolat fumant !... J'eus quand même pitié de lui, ce qui me fit lui demander :

P. — Alors, par quoi on commence ?

O. — Ah, enfin !... Par ce dont on avait parlé samedi dernier !

P. — Et de quoi avions nous parlé, demandais-je innocemment ?

O. — Tu m'avais dit que tu me montrerais comment on fait pour créer le mouvement.

Déplacement de gauche à droite

P. — On va donc commencer par le plus simple, c'est-à-dire par le déplacement horizontal d'un motif de la dimension d'un carreau, comme une balle par exemple !... Cela te convient ?

O. — Oui, tout à fait !

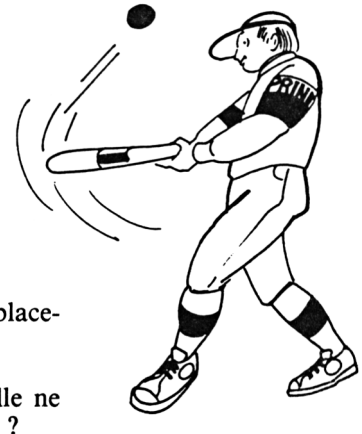
P. — Te souviens-tu comment on fait pour dessiner une ligne horizontale ?

O. — Bien sûr !

P. — Eh bien, l'idée de base est la même !... Tu écris une boucle qui fait avancer la balle de case en case depuis le bord gauche de l'écran jusqu'au bord droit. Simplement, pour éviter d'obtenir une suite continue de balles, tu effaces après chaque déplacement de la balle l'ancien emplacement qu'elle occupait !

O. — J'ai compris !... C'est effectivement tout simple !... Voici donc un petit programme qui devrait aller :

```
10 FOR C=0 TO 31
20 PRINT AT 10,C;"O"
30 PRINT AT 10,C-1;"  "
40 NEXT C
```



P. — Pas mal pour un début !... Je vois que ta ligne 30 est celle qui efface l'emplacement précédent de la balle !... Et qu'est-ce que cela donne ?

O. — Ça donne que la balle se déplace bien de gauche à droite, mais qu'elle ne s'efface pas une fois arrivée sur le bord droit de l'écran !... Cela provient de quoi ?

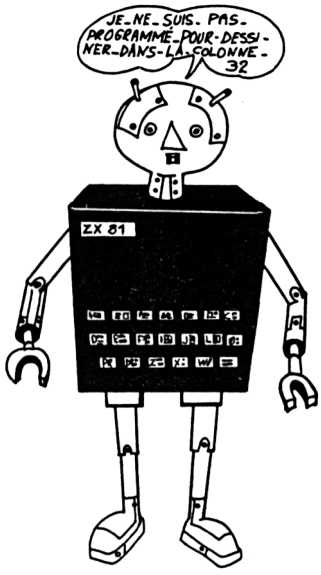
P. — De ce que tu n'effaces que jusqu'à la 30e colonne, puisque tu le fais à la C – 1ème et que C vaut 31 au maximum. Augmente donc la valeur de C jusqu'à 32 et tu verras que tu effaceras alors la balle lorsqu'elle arrive à droite.

O. — Bonne idée !... Voilà, c'est fait !

```
10 FOR C=0 TO 32
```

P. — Et cela marche ?

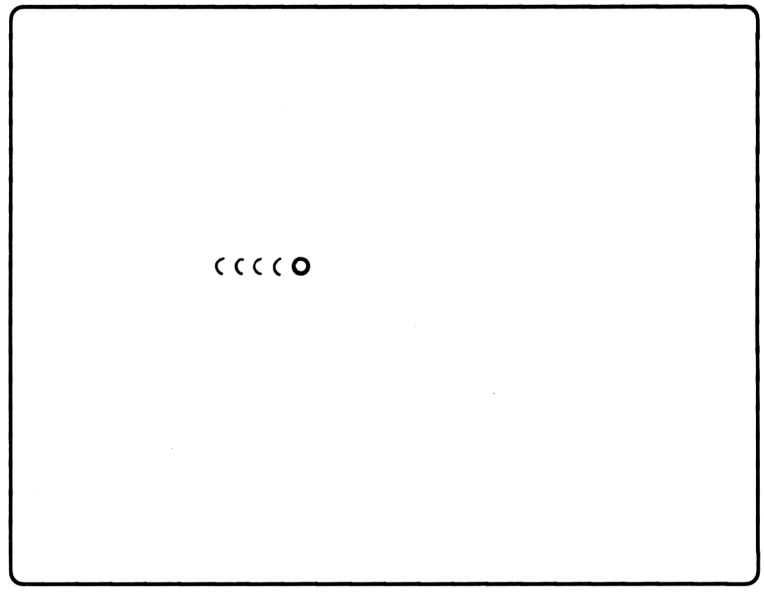
O. — Toujours pas !... Et, en plus, j'obtiens un message d'erreur B/20 !... Qu'est-ce qu'il se passe à ton avis ?



P. — Il se passe qu'à la ligne 20 tu demandes à l'ordinateur de dessiner une balle dans la colonne 32 lorsque C vaut 32, et cela, il ne peut le faire !... Il faut donc le lui interdire avec une ligne du genre :

```
20 IF C<32 THEN PRINT AT 10,C;
"O"
```

O. — Tu as raison, ça marche !... Mais c'est pas encore parfait !... Par moments, on dirait qu'il y a deux balles sur l'écran !



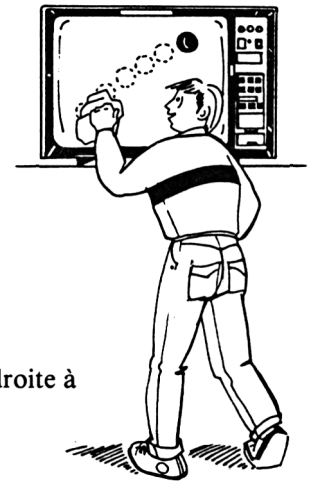
P. — C'est normal, car tu inscris d'abord la balle dans une nouvelle case, avant de l'effacer dans la case précédente. Ainsi, il y a deux balles visibles sur l'écran pendant un temps très bref, à chaque parcours de la boucle !

O. — Donc, si j'ai bien compris, il suffit d'effacer avant d'écrire pour faire disparaître ce phénomène !

P. — Exactement !

O. — Voilà donc le programme dans le bon ordre :

```
10 FOR C=0 TO 32
20 PRINT AT 10,C-1;" "
30 IF C<32 THEN PRINT AT 10,C;
"O"
40 NEXT C
```



P. — Eh bien, tu vois que tu y es arrivé !... Bravo !

Déplacement de droite à gauche

P. — Puisque tu as compris, il ne te reste plus qu'à faire la même chose de droite à gauche !

O. — C'est comme si c'était déjà fait !

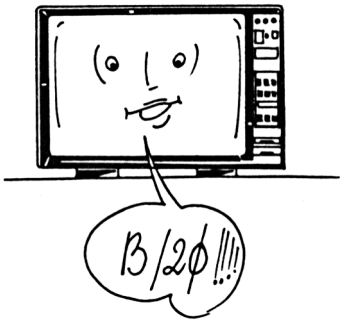
P. — Attends, on va compliquer un peu !... Qu'est-ce que tu dirais d'essayer cela sur un sous-marin de trois carreaux de longueur ?

O. — D'accord !... Comme cela, ça changera un peu !... Voici donc le petit programme qui devrait permettre ce déplacement de droite à gauche :

```
10 FOR C=31 TO -1 STEP -1
20 PRINT AT 10,C+1;" "
30 IF C>-1 THEN PRINT AT 10,C;
"▟▙"
40 NEXT C
```

P. — Si je comprends bien, tu as tout simplement copié ce programme sur le précédent, en remplaçant la valeur limite 32 par −1 et la colonne d'effacement C−1 par C+1 !... Penses-tu que cela va marcher ?

O. — Il n'y a pas de raisons !... Bouge pas, j'essaye !... Tiens, il n'y a rien qui apparaît sur l'écran !



P. — Tu n'aurais pas un message d'erreur, par hasard ?

O. — Si !... B/20 !

P. — Il t'indique qu'à la ligne 20 tu demandes à l'ordinateur d'imprimer un espace dans la colonne 32 quand C vaut 31 !... Il te faut empêcher cela avec une ligne d'instruction du genre :

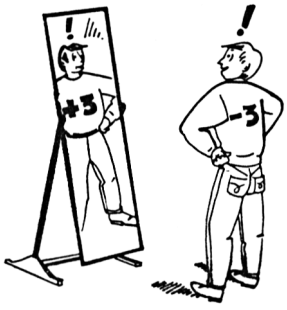
```
20 IF C<31 THEN PRINT AT 10,C+
1;" "
```

O. — Pourtant, dans l'exemple précédent, il n'avait pas été utile d'écrire cela ?

P. — Effectivement !... Mais ce que tu ne sais pas, c'est que le bord gauche de l'écran ne se comporte pas comme le bord droit. Autant le bord gauche est tolérant quant tu lui demandes par mégarde d'écrire dans une colonne inférieure à 0, autant le bord droit est intolérant quand tu lui demandes d'aller au-delà de 31.

O. — Qu'est-ce que tu entends par écrire par mégarde dans une colonne inférieure à Ø ?

P. — Je veux dire que dans le programme de l'exemple précédent, tu demandais à la ligne 2Ø d'imprimer un espace dans la colonne −1 quand C valait Ø !... Cela ne s'est pas traduit par un défaut très visible, mais, pour un œil exercé, il y était !... Ce qu'il faut savoir, c'est que le bord gauche de l'écran, et uniquement celui-là, se comporte comme un miroir. Si tu demandes à l'ordinateur d'afficher quelque chose dans la colonne −3 par exemple, il va tout simplement l'afficher dans la colonne 3 car il ne tient pas compte du signe.



O. — Ça y est, je viens de comprendre !... Encore une fois, c'est plein d'astuces qu'il faut que je découvre !

P. — Bien !... Te considères-tu satisfait de ton travail ?

O. — Pas précisément !... Tout d'abord, l'arrière du sous-marin ne s'efface pas et, en plus, il s'inscrit deux rectangles noirs à la gauche de l'écran, sur la ligne du dessous.

P. — Commençons par le premier point !... Tu n'effaces pas à l'arrière du sous-marin parce que tu n'effectues pas l'effacement au bon endroit, voilà tout !... Si tu analyses ce que tu as écris, tu vas te rendre compte que tu demandes à l'ordinateur d'effacer la première case du sous-marin !... Cela se voit d'ailleurs à l'écran car l'avant du sous-marin semble clignoter.

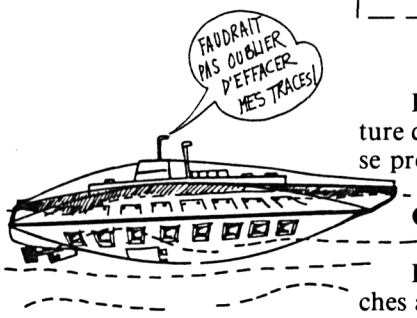
O. — Ça, ça m'apprendra à recopier bêtement un programme !... Il est évident que je dois effacer 3 cases plus loin, c'est-à-dire à C+4, comme cela :

```
20 IF C<31 THEN PRINT AT 10,C+
4;"  "
```

P. — Et alors ?

O. — Alors, ça ne marche plus !... Encore une fois, j'ai le message d'erreur B/2Ø !... Ah, j'ai trouvé !... Il fallait également que je modifie la valeur de C à partir de laquelle l'effacement doit commencer !... Cela va donc donner :

```
20 IF C<28 THEN PRINT AT 10,C+
4;"  "
```



P. — Bravo !... Maintenant, passons au second point que tu avais constaté : l'écriture des deux rectangles sur la ligne en dessous du sous-marin. As-tu remarqué quand cela se produisait ?

O. — Oui, au tout début de chaque déplacement !

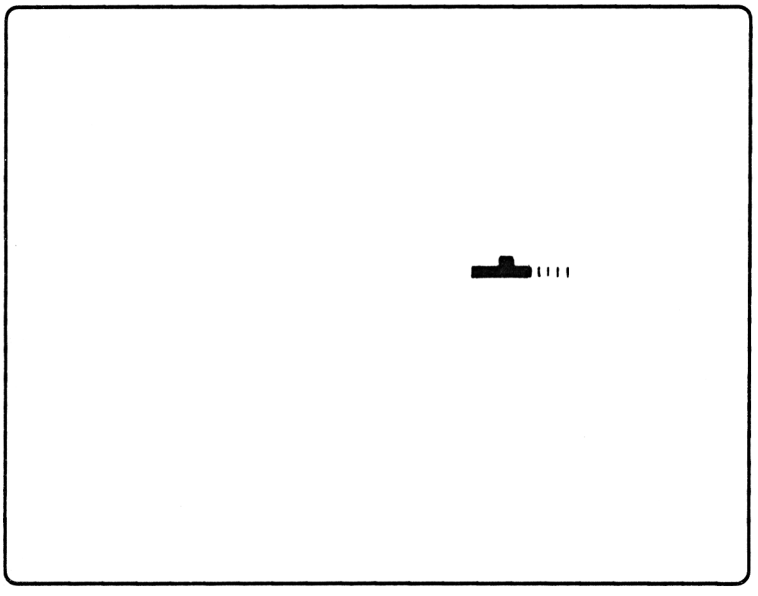
P. — D'après ce que tu me dis, cela doit provenir de ta ligne 3Ø avec laquelle tu cherches à imprimer un sous-marin de trois cases de longueur lorsque C vaut 31. Comme tu ne disposes que d'un carreau pour cela, celui de la colonne 31, la mémoire d'affichage reçoit l'ordre de déborder sur la ligne du dessous et, bien entendu, au début de cette ligne, c'est-à-dire à gauche de l'écran.

O. — Qu'est-ce que c'est compliqué !... Tu vois une solution à ce problème ?

P. — Oui !... Il est inutile de faire démarrer ta boucle à C = 31 !... Essaye donc avec C = 29 !

```
10 FOR C=29 TO -1 STEP -1
```

O. — Gagné, ça marche !... Il ne reste plus qu'à résoudre le problème que pose l'effacement du sous-marin lorsqu'il arrive sur le bord gauche de l'écran !



P. — Oui, mais avant, je vais te montrer comment on peut simplifier l'écriture du programme que nous venons péniblement d'écrire tous les deux !

O. — Oh non, ne me dis pas qu'il faut tout recommencer !

P. — Non !... D'ailleurs, c'est moi qui vais le faire !... Voilà, j'ai terminé !...

```
10 FOR C=28 TO 0 STEP -1
20 IF C>=0 THEN PRINT AT 10,C;
"▬▬ "
30 NEXT C
```

O. — Effectivement, le résultat est le même !... Tu peux m'expliquer pourquoi ?

P. — Comme tu peux t'en rendre compte, j'ai supprimé l'ancienne ligne 20 qui servait à l'effacement et je l'ai remplacée par un espace situé à l'arrière du sous-marin. Ainsi, comme le sous-marin en se déplaçant laisse derrière lui un sillage qui n'est rien d'autre que le dernier motif imprimé, avec un espace, il ne laisse plus de traces.

O. — Pas mal !... C'est donc pour ça que tu as ramené la valeur de début de boucle à C = 28 au lieu de C = 29 !... Mais pourquoi avoir remplacé la valeur de fin de boucle de −1 par 0 ?

P. — Parce que tu effaces avec la même ligne que celle qui imprime, alors que, tout à l'heure, tu effaçais avant d'imprimer !... Il n'y a donc pas lieu d'aller en deçà de la valeur de la limite que constitue la colonne 0 !

O. — C'est vrai !... Alors, je l'adopte ton programme !... Tu me montres maintenant comment on fait pour effacer ?

P. — Pour effacer, il existe deux méthodes. L'une qui consiste à effacer d'un bloc et l'autre d'une façon progressive.

O. — J'imagine que l'effacement progressif est assez compliqué ?

P. — Oh oui !

O. — Alors, restons dans le simple !... Montre-moi comment on efface d'un seul coup !

P. — Cela s'obtient en écrivant une ligne supplémentaire que tu peux placer à l'intérieur ou à l'extérieur de la boucle, selon ton bon vouloir !... A l'intérieur, elle s'écrit comme ceci :

```
25 IF C=Ø THEN PRINT AT 1Ø,C;"
  "
```

P. — ... alors qu'à l'extérieur, il faut l'écrire de manière légèrement différente en remplaçant C = Ø par C = – 1 et en commençant l'effacement à la colonne C + 1 au lieu de C :

```
40 IF C=-1 THEN PRINT AT 1Ø,C+
1;"    "
```

O. — Alors là, je ne comprends pas pourquoi, puisque C varie de 28 à Ø !

P. — Pas exactement !... En réalité, lorsque la boucle est parcourue pour la dernière fois, C vaut alors Ø jusqu'à la ligne 3Ø. Arrivé à cette ligne, C prend alors la valeur qu'il croit être la suivante, c'est-à-dire – 1, et comme cette valeur n'entre pas dans les limites de la ligne 1Ø, C conserve cette valeur tout au long du reste du programme. C'est pourquoi il faut effectuer l'effacement avec C = – 1, puisque cela se passe en dehors de la boucle !

O. — Encore un nouveau « truc » qu'il faut que je me souvienne !

P. — Comme tu dis !... Maintenant, si tu y tiens, je peux te montrer une méthode d'effacement très expéditive ?

O. — Dis toujours !

P. — Elle consiste à effacer la totalité de l'écran en une seule fois !... Seulement là, tout y passe !... Le sous-marin et le reste !... Cette instruction du type marteau-pilon s'appelle CLS !

O. — Et que veut-elle dire ?

P. — Il s'agit de l'anglais « Clear Screen », ce qui veut dire : effacement de l'écran !

O. — En voilà une facile à se rappeler !... Et elle s'emploie comment ?

P. — Tout simplement comme cela :

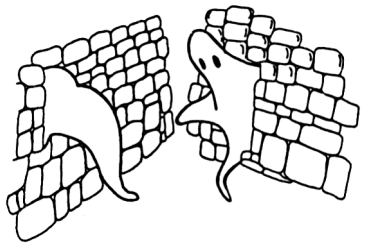
```
40 CLS
```

O. — Tu as raison, on ne fait pas plus court !... Et comment ça marche ?

P. — En mettant toutes les cases de la mémoire d'affichage à zéro, ce qui correspond, si tu regardes ton manuel d'utilisation à la page 181, à l'espace blanc.

O. — Eh bien, j'aurais jamais cru que déplacer un sous-marin d'un bord à l'autre de l'écran pouvait poser tant de problèmes !

P. — N'est-ce pas ?... Et tu n'as pas encore tout vu !... Laisse-moi un peu te parler de l'apparition et de la disparition progressive sur l'écran.



Apparition et disparition progressives sur l'écran

O. — Je sens qu'on va s'amuser !

P. — Comme tu dis !... En fait, si je te montre cela, c'est pour que tu prennes conscience des contraintes que posent l'apparition et la disparition progressives d'un mobile, même si au départ tu penses que cela est plus réaliste !... Mais, trève de bavardages, passons aux actes !

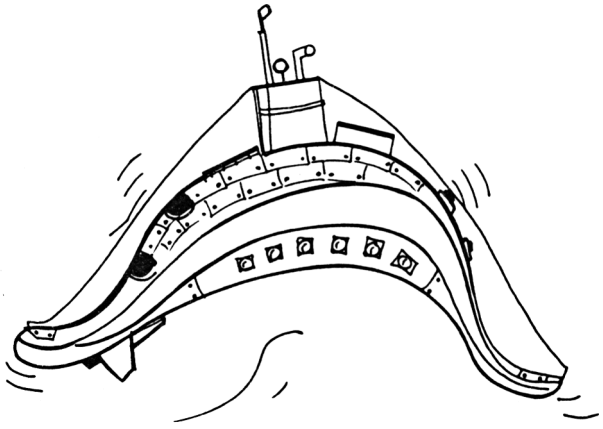
O. — Par quoi on commence ?

P. — Par le début, c'est-à-dire par l'apparition progressive !... Lorsque tu veux faire apparaître un mobile de manière progressive, il te faut écrire autant de lignes d'instructions que ce mobile comporte de carreaux de longueur, plus une ligne supplémentaire pour l'espace blanc d'effacement.

O. — Alors, je devine à quoi est liée la première contrainte. Cela va occuper trop de place dans la mémoire !

P. — Gagné !... D'ailleurs, voici le programme :

```
10 FOR C=31 TO 0 STEP -1
20 PRINT AT 10,C;"▬"
30 IF C<31 THEN PRINT AT 10,C+
1;"▟"
40 IF C<30 THEN PRINT AT 10,C+
2;"▬"
50 IF C<29 THEN PRINT AT 10,C+
3;" "
60 NEXT C
```



O. — Qu'est-ce qu'il est compliqué !

P. — Moi, je dirais plutôt qu'il est long !... Tu te rends compte, une ligne d'instruction par carreau de longueur !... Heureusement que tu n'as pas cherché à dessiner un sous-marin atomique de 7 à 8 carreaux de longueur !... Vérifie quand même si ce programme fonctionne bien !

O. — Oui, tout va bien !... L'apparition du sous-marin se fait très progressivement !... Par contre, quand il se déplace sur l'écran, on a l'impression qu'il avance comme un ver de terre étant donné qu'il s'allonge et se raccourcit tout le temps !

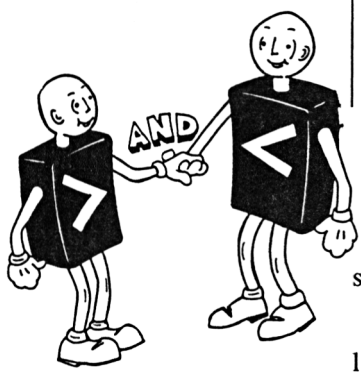
P. — Là, tu viens de mettre le doigt sur le second défaut de la méthode : la progression est devenue trop lente et saccadée !... Trop lente parce qu'il y a trop d'instructions dans la boucle ; trop saccadée parce que le sous-marin est découpé en petits morceaux dont aucun n'avance en même temps que les autres !... Tu verras néanmoins que, sous certaines conditions, cela permet d'obtenir des résultats intéressants. Je t'en parlerai plus tard !

O. — Bon !... Il nous reste donc à voir comment on fait pour effacer progressivement !

P. — L'effacement progressif, tu vas l'obtenir en permettant à la variable C de diminuer jusqu'à la valeur −3, tout en t'assurant que chaque ligne d'instruction ne puisse rien tenter d'écrire en deçà de la colonne Ø !

O. — J'ai compris !... Laisse-moi m'occuper de cela !... Voici déjà pour les deux premières lignes :

```
10 FOR C=31 TO -3 STEP -1
20 IF C>-1 THEN PRINT AT 10,C;
"▬"
```



P. — Bien !... Et pourquoi ne pas avoir terminé les autres ?

O. — Pour tout t'avouer, je ne vois pas comment je peux fixer à la fois une limite supérieure et une limite inférieure à une instructions IF ?

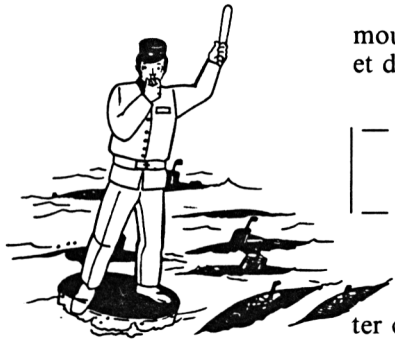
P. — Tu as raison, ce n'est pas évident à première vue. Il te faut employer pour cela l'intruction AND en écrivant ta ligne d'instruction ainsi : IF C >... AND C <... THEN PRINT AT... !... Ainsi, l'instruction n'est exécutée que si la valeur de C est comprise entre ces deux limites !

O. — C'est vrai, on avait déjà utilisé cette instruction pour les coordonnées des cloisons du labyrinthe, mais comme il s'agissait de deux variables différentes, je ne pensais pas qu'on pouvait s'en servir pour deux valeurs différentes de la même variable !... Alors, voilà la suite du programme :

```
30 IF C>-2 AND C<31 THEN PRINT
AT 10,C+1;"▬"
40 IF C>-3 AND C<30 THEN PRINT
AT 10,C+2;"▬"
50 IF C<29 THEN PRINT AT 10,C+
3;" "
60 NEXT C
```

P. — Bravo !... Puisqu'on en est là, je vais te montrer comment tu peux réaliser un mouvement perpétuel de passage de sous-marins !... Ajoute donc cette ligne d'instruction et dis-moi ce que tu en penses :

```
70 GOTO 10
```



O. — Terrible !... Ça n'arrête pas !... Au fait dis-donc, comment fait-on pour arrêter cela ?

P. — On appuie sur la touche [BREAK] qui interrompt aussitôt le déroulement du programme. Il te suffit ensuite d'appuyer sur [NEW LINE] pour obtenir à nouveau ta liste d'instructions.

O. — Effectivement, ça marche !... « Break » en anglais, ça ne veut pas dire casser, interrompre ?

P. — Si justement !

O. — Maintenant, pour en terminer avec cela, j'aimerais que tu m'expliques dans quelles conditions un tel programme peut présenter de l'intérêt !

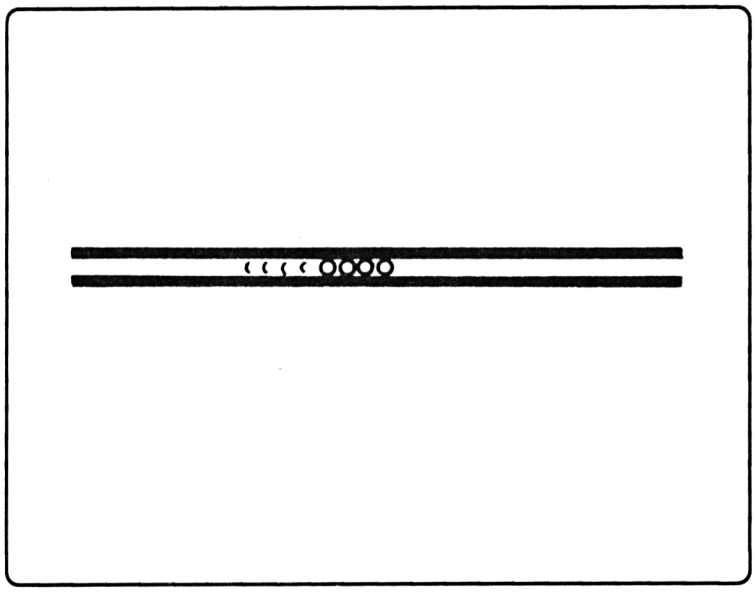
P. — Tu as raison, finissons-en !... Cette façon de procéder n'est valable que si le dessin de ce que tu veux représenter emploie toujours le même motif. Dans ces conditions, l'écriture se simplifie à deux lignes d'instructions, quelle que soit la longueur du dessin.

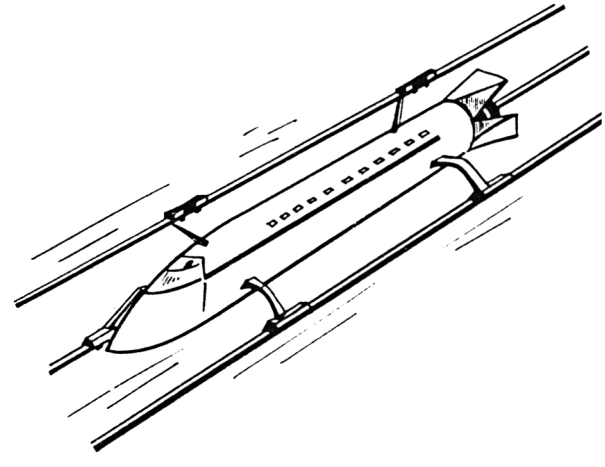
O. — Ça, ça me paraît plus intéressant !... Tu me fais voir !

P. — Tiens, voici l'exemple d'un train futuriste qui circule dans un tunnel magnétique :

```
10 FOR C=0 TO 35
20 IF C<32 THEN PRINT AT 9,C;"
";AT 10,C;"O";AT 11,C;"
30 IF C>3 THEN PRINT AT 10,C-4
;"  "
40 NEXT C
50 GOTO 10
```

O. — D'après ton programme, je vois que le train a 4 carreaux de longueur !





P. — Exact !... As-tu vu également comment j'ai fait pour dessiner le tunnel ?... En même temps que le premier passage du train !... Toutefois, je ne te recommande pas de faire comme cela, parce que tu ralentis trop la vitesse du train en réimprimant le tunnel à chaque passage !

O. — Ah bon !... C'est vrai que cela ne se voit pas !... Maintenant qu'on a fini avec cela, j'aimerais te parler d'un petit problème qui me préoccupe !

P. — Vas-y, je t'écoute !

Déplacements en sens inverse

O. — Est-ce que tu crois qu'il est possible de faire évoluer en même temps deux engins sur l'écran, si les deux engins se déplacent en sens inverse l'un par rapport à l'autre ?

P. — Bien sûr !... Pourquoi pensais-tu que cela ne le serait pas ?

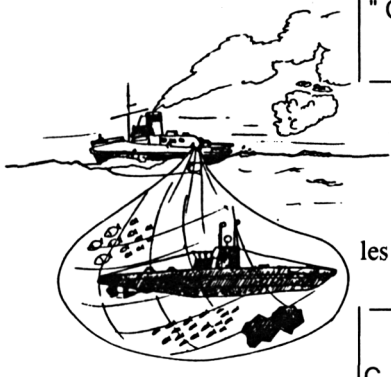
O. — Parce que, quand tu veux déplacer deux objets, il faut placer leurs lignes d'impression dans une même boucle !... Or la boucle est conçue pour les faire aller de gauche à droite, ou de droite à gauche, mais pas dans les deux sens en même temps.

P. — C'est vrai qu'en théorie, tu as raison !... Cependant, je vais te montrer comment, avec une boucle qui est prévue pour te donner normalement un déplacement de gauche à droite, tu peux obtenir exactement l'effet inverse.

O. — Ah, c'est intéressant !... Tu me fais voir ça avec une balle ?

P. — Ok !... J'écris d'abord le programme normal qu'on a déjà employé :

```
10 FOR C=0 TO 32
20 IF C>0 THEN PRINT AT 10,C-1
;"  "
30 IF C<32 THEN PRINT AT 10,C;
"O"
40 NEXT C
```



O. — Ça, je reconnais !... C'est la version améliorée de la page 95 !

P. — Exact !... Et c'est à partir de ce programme que je vais travailler, en modifiant les lignes 20 et 30 comme ceci :

```
20 IF C>0 THEN PRINT AT 10,32-
C;"  "
30 IF C<32 THEN PRINT AT 10,31
-C;"O"
```

O. — Effectivement, on obtient un déplacement en sens inverse de celui prévu par la boucle !... Bravo !... A mon tour d'essayer !

P. — Avec quoi vas-tu essayer ?

O. — Avec un bateau et un sous-marin !... Le bateau circulera de gauche à droite en surface et le sous-marin de droite à gauche sous l'eau.

P. — Bonne idée !... Cela devrait être du meilleur effet, ces deux navires voguant en sens inverse l'un de l'autre !... Par quoi tu commences ?

O. — Par le bateau de surface !... Voici son programme :

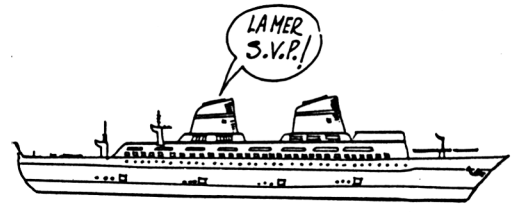
```
10 FOR C=0 TO 27
20 PRINT AT 0,C;"  -▪▙"
30 PRINT AT 1,C;"  ▬▬▬"
40 NEXT C
```

P. — Très bien, mais il te reste à dessiner la mer !

O. — Zut, j'avais oublié !... Si je me souviens bien, il faut dessiner la mer avant le bateau !

P. — Exact !

O. — Alors, voici le programme de la mer :




```
5 FOR C=Ø TO 31
6 PRINT AT 1,C;"-"
7 NEXT C
```

P. — Parfait !

O. — Tiens, il y a un défaut !... La mer s'efface derrière le bateau au fur et à mesure qu'il avance !

P. — Cela provient de ce que tu effaces le sillage de ton bateau à l'aide d'un espace blanc. Dans ce cas, il vaudrait mieux effacer à l'aide d'un signe moins car, ainsi, cela te redessinerait la mer derrière le bateau.

O. — Bonne idée !... Je modifie tout de suite ma ligne 3Ø d'instruction :

```
3Ø PRINT AT 1,C;"-━━━━━━"
```

O. — ... et voilà, ça marche !... En se déplaçant le bateau n'efface plus la mer !... Génial, le coup du pointillé !

P. — Ceci dit, comment vas-tu effacer le bateau en bordure d'écran ?

O. — Tout simplement avec CLS comme tu me l'as montré !... Voilà ce que ça donne :

```
5Ø CLS
```

P. — Es-tu satisfait du résultat obtenu ?

O. — Tout à fait !

P. — Dans ces conditions, je te suggère d'écrire la ligne d'instruction qui permet de faire réapparaître le bateau sur le bord gauche de l'écran, après qu'il ait disparu du bord droit.

O. — Facile !... La voici :

```
6Ø GOTO 5
```

O. — ... Mince alors !... Avec l'instruction CLS, j'efface tout, le bateau comme la mer !... Ce qui fait que l'ordinateur doit redessiner la mer avant de faire réapparaître le

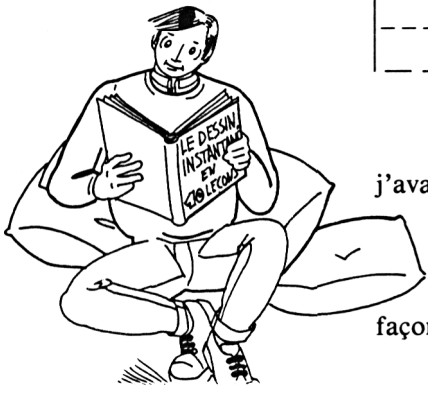
bateau, et ça, ça ne fait pas très beau !... Je vais devoir effacer autrement qu'avec CLS !

P. — Pas forcément !... Le problème, c'est le temps que mettent les lignes d'instruction 5, 6 et 7 pour dessiner la mer !... Si tu pouvais la dessiner instantanément, tu pourrais continuer à employer CLS !

O. — Et c'est possible de dessiner instantanément la mer ?

P. — Bien sûr !... Tu l'as même déjà fait pour autre chose et tu ne t'en souviens plus !... Essaye donc ceci :

```
5 PRINT AT 1,0;"--------------
-------------------"
```



O. — Ça marche !... C'est tout juste si l'on voit l'écran clignoter !... Où est-ce que j'avais déjà écrit quelque chose comme ça ?

P. — Quand tu dessinais ta route avec l'instruction SCROLL !

O. — Ah oui, je me souviens maintenant !... La route s'imprimait ligne par ligne de façon quasi instantanée !

P. — Maintenant que tu as tout compris, si nous passions à la suite, c'est-à-dire au sous-marin !

O. — C'est vrai, je l'avais presque oublié celui-là !... Au fait, sa longueur est plus courte d'un carreau que celle du bateau !

P. — Oui !... Cela te laisse le choix pour le faire apparaître un carreau plus tard ou disparaître un carreau plus tôt !

O. — Qu'est-ce que tu me conseilles ?

P. — Fais-le disparaître un carreau plus tôt, la partie intéressante se situant avant sa rencontre avec le bateau !

O. — D'accord !... Alors voici comment j'envisage son déplacement en sens inverse :

```
10 FOR C=0 TO 27
20 PRINT AT 10,28-C;"▟▙  "
30 NEXT C
40 CLS
50 GOTO 10
```

P. — Je vois que tu as choisi les mêmes valeurs limites pour la variables C que celles du bateau !

O. — C'est exprès !... C'est pour pouvoir ensuite tout incorporer dans la même boucle !... Ce que d'ailleurs, je vais faire tout de suite :

```
5 PRINT AT 1,0;"--------------
-------------------"
10 FOR C=0 TO 27
```

```
20 PRINT AT 0,C;"  -▪▙"
30 PRINT AT 1,C;"-▬▬▬"
40 PRINT AT 10,28-C;"▬▟▬  "
50 NEXT C
60 CLS
70 GOTO 5
```

P. — Bravo !... Mes félicitations !... Pour la peine, je vais t'enseigner un « truc ».

O. — Chouette !... Qu'est-ce que c'est ?

P. — Cela concerne la position de l'instruction CLS !... Telle qu'elle est placée, il y a la ligne d'instruction 70 qui la sépare de la ligne 5 !

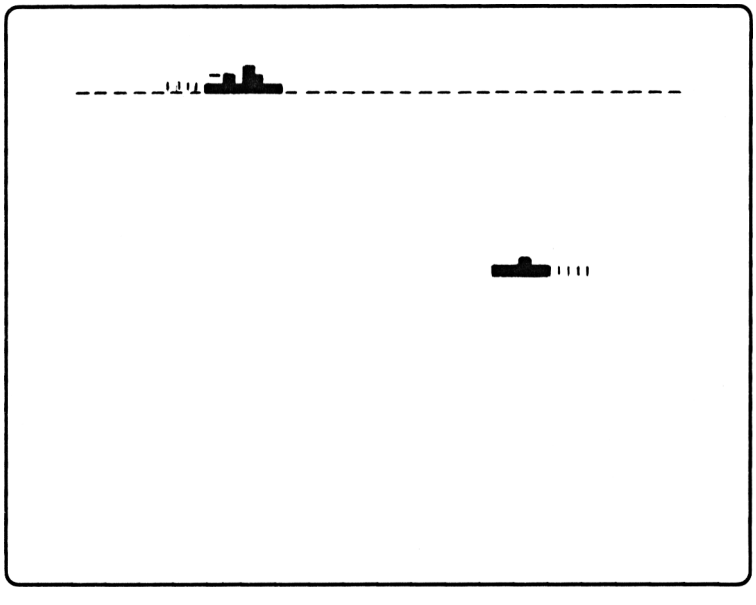
O. — Et après ?

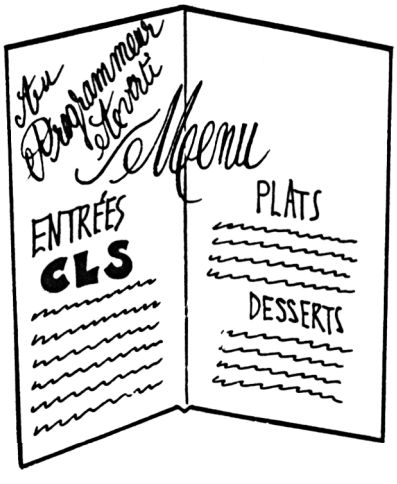
P. — Après, l'écran reste vide pendant la durée de la ligne 70, ce qui accentue le phénomène de clignotement de la mer !... L'astuce consiste à placer l'instruction CLS en début de programme, juste avant la ligne d'impression de la mer. Ainsi, tu réimprimes juste après avoir effacé, ce qui n'est pratiquement pas perceptible !

O. — Effectivement, c'est astucieux !... Attends, je vais modifier le programme en conséquence :

```
10 CLS
20 PRINT AT 1,0;"-------------
-------------------"
30 FOR C=0 TO 27
40 PRINT AT 0,C;"  -▪▙"
50 PRINT AT 1,C;"-▬▬▬"
60 PRINT AT 10,28-C;"▬▟▬  "
70 NEXT C
80 GOTO 10
```

P. — Alors n'est-ce pas mieux ainsi ?





O. — Si fait, si fait !... C'est à peine si on s'aperçoit que la mer s'efface par moments.

Les déplacements verticaux

P. — Bien !... Maintenant que tu maîtrises bien les déplacements horizontaux, serais-tu tenté par ceux qui sont verticaux ?

O. — Bien sûr !... D'autant plus que je vais pouvoir écrire les programmes tout seul !

P. — Alors, par quoi commençons-nous ?

O. — Par le déplacement de haut en bas !

P. — Ok !... Et que vas-tu choisir comme engin pour faire cela ?

O. — Une soucoupe volante !... Ça fait un moment que je rêve d'en faire voler une !... Voici déjà son programme de descente :

```
10 FOR L=0 TO 22
20 IF L>0 THEN PRINT AT L-1,15
;"    "
30 IF L<22 THEN PRINT AT L,15;
"▬▬"
40 NEXT L
```

P. — Très chouette ta soucoupe qui descend en feuille morte !... Et si tu la faisais atterrir au lieu de la faire disparaître en bas de l'écran ?

O. — Très bonne idée !... Le temps de dessiner le sol et de modifier le programme de descente et c'est fait !... Voici déjà pour le sol :

```
6 FOR C=0 TO 31
7 PRINT AT 21,C;"▬"
8 NEXT C
```

O. — ... et voici pour l'atterrissage en douceur :

```
10 FOR L=0 TO 20
20 IF L>0 THEN PRINT AT L-1,15
;"    "
30 PRINT AT L,15;"▬▬"
40 NEXT L
```

P. — Très bien, dis donc !... Je remarque que tu as modifié l'une des valeurs limites de la boucle, et que tu as supprimé toute condition à la ligne d'instruction 30.

O. — Oui !... Comme je fais atterrir la soucoupe sur un sol placé à la ligne 21, il n'y a pas lieu de faire aller L au-delà de la ligne 20, puisque cette ligne est celle qu'occupera la

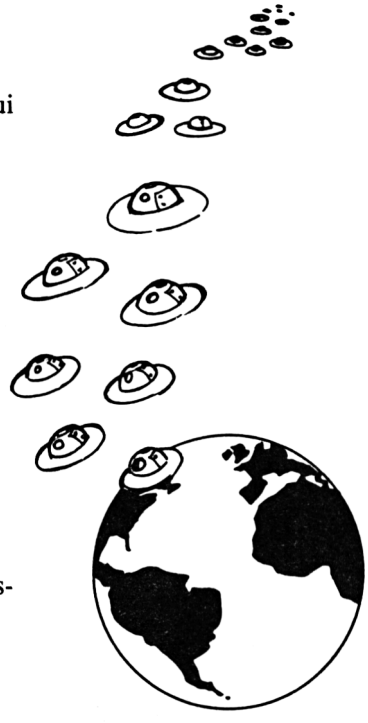
soucoupe au sol. Du même coup, il n'est pas utile d'interdire l'impression de la soucoupe lorsque L vaut 22, puisque cette valeur ne peut plus être atteinte.

P. — Cet atterrissage me donne encore une idée !... Tu ne veux pas t'amuser à faire atterrir toute une flotille de soucoupes ?

O. — Si, cela me plairait !... Tu me montres comment il faut établir un programme capable de faire cela ?

P. — Tout de suite !... Il te suffit de placer ta boucle d'atterrissage dans une plus grande boucle qui détermine le nombre de soucoupes qui doivent atterrir, comme cela :

```
 9 FOR N=1 TO 29 STEP 4
50 NEXT N
```



P. — ... s'en oublier, bien sûr, de modifier les lignes d'instruction 20 et 30 qui deviennent :

```
 20 IF L>0 THEN PRINT AT L-1,N;
"  "
 30 PRINT AT L,N;"▇▇"
```

O. — Super !... Tiens, voilà la saturation de la mémoire qui recommence !

P. — Où se produit-elle ?

O. — A la descente de la septième soucoupe !

P. — Bon !... Il suffit donc d'en limiter le nombre à six, en modifiant la ligne d'instruction 9 comme suit :

```
 9 FOR N=1 TO 21 STEP 4
```

O. — Ok, c'est bon !... Il ne reste plus qu'à faire décoller tout ça maintenant !

P. — Excellente idée !... Ainsi, cela va correspondre au déplacement de bas en haut que tu voulais étudier !... Dans ces conditions, je te conseille d'écrire ton programme de décollage à la suite de celui qu'on vient d'écrire, de façon que tu conserves toutes les soucoupes sur l'écran avant leur envol. Cela t'évitera d'avoir à les redessiner !

O. — Bien « Chef » !... Le temps d'écrire le programme... Ça y est, je suis prêt à décoller :

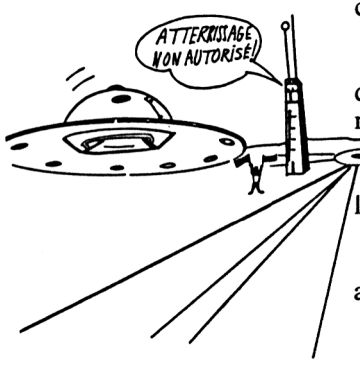
```
 60 FOR N=1 TO 21 STEP 4
 70 FOR L=20 TO -1 STEP -1
 80 IF L<20 THEN PRINT AT L+1,N
;"  "
 90 IF L>-1 THEN PRINT AT L,N;"
▇▇"
100 NEXT L
110 NEXT N
```

P. — Et alors ?

O. — Devines ce qui se passe ?

P. — Je parie que tu as une saturation de la mémoire.

O. — Eh oui !... Et en plus, cela se produit pendant la descente de la cinquième soucoupe !... Je ne vais quand même pas me limiter à quatre soucoupes ?



P. — Je suis bien d'accord avec toi !... On va refaire le même petit exercice de style que celui qu'on avait exécuté sur le dessin du labyrinthe afin d'économiser de la place en mémoire.

O. — Ah oui, je me souviens !... C'est quand on avait remplacé les nombres par des lettres !

P. — Exactement !... Toutefois, il est vraisemblable qu'on n'arrivera pas à faire atterrir six soucoupes. Je te propose donc de limiter leur nombre à cinq !

O. — D'accord !... Pour le programme, tu ne veux pas m'aider ?

P. — Si, bien sûr !... Je vais te définir tes valeurs littérales !... En examinant ton programme, je vois que tu emploies le Ø, le 1, le 4, le 16, le 2Ø et le 21. Voici donc ces valeurs :

```
 1 LET Z=PI-PI
 2 LET U=SGN PI
 3 LET Q=INT (PI+U)
 4 LET S=Q*Q
 5 LET V=S+Q
```

O. — Merci !... J'avoue que je n'aurais jamais pensé à définir le chiffre 4 comme tu l'as fait !... Par contre, à quoi correspond l'instruction SGN ?

P. — L'instruction SGN indique le signe d'une quantité. Elle est donc égale à +1, −1 ou Ø, selon que la quantité est positive, négative ou nulle.

O. — En voilà au moins une que je retiendrai facilement, d'autant plus que l'abréviation laisse deviner qu'il s'agit du mot « signe » !

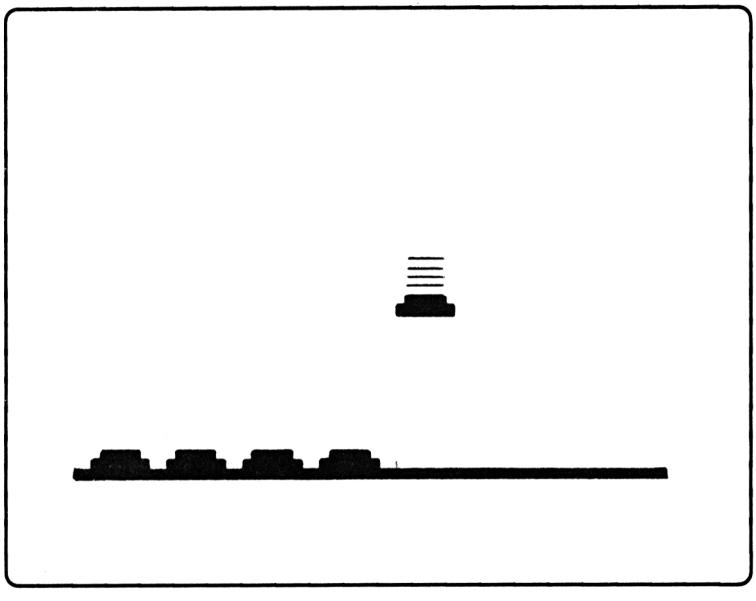
P. — C'est ce qu'on appelle la mnémonique des instructions BASIC !... En fait, elles sont toutes écrites de façon à évoquer ce à quoi elles servent, mais dans une langue qui est l'anglais. Ce n'est que lorsque le mot est le même dans les deux langues que la mnémonique sert !

O. — Bien !... Il ne me reste plus qu'à reprendre tout le programme en remplaçant les nombres par des lettres, et en limitant le nombre de soucoupes à cinq :

```
 6 FOR C=Z TO V
 7 PRINT AT V+U,C;"■"
 8 NEXT C
 9 FOR N=U TO S+U STEP Q
1Ø FOR L=Z TO V
2Ø IF L>Z THEN PRINT AT L-U,N;
"  "
3Ø PRINT AT L,N;"■■"
4Ø NEXT L
5Ø NEXT N
6Ø FOR N=U TO S+U STEP Q
7Ø FOR L=V TO -U STEP -U
```

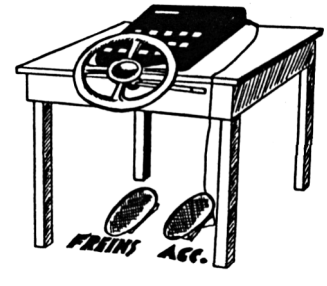
```
 80 IF L<V THEN PRINT AT L+U,N;
"    "
 90 IF L>-U THEN PRINT AT L,N;"
   "
100 NEXT L
110 NEXT C
```

P. — Et ça fonctionne !... Bravo, tu as eu raison de cette sacrée mémoire !



O. — Oui !... Quoique je constate à travers cet exemple que si cette forme d'écriture permet d'économiser de la place dans la mémoire de programmation, on n'en gagne que très peu dans la mémoire d'affichage !

P. — Exact !... Bien souvent, le jeu n'en vaut pas la chandelle quand ce n'est que la mémoire d'affichage que tu veux désaturer !... Par contre, c'est très efficace pour la mémoire de programmation !



Comment modifier la vitesse

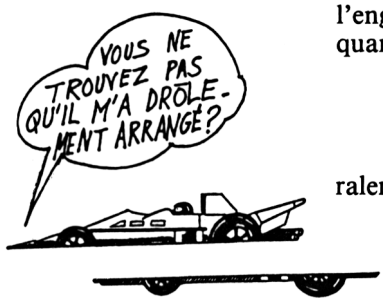
O. — Tous ces atterrissages et décollages viennent de me faire penser à quelque chose !

P. — A quoi donc ?

O. — N'y a-t-il pas un moyen qui permette de faire varier la vitesse de déplacement d'un engin ?

P. — Si, bien sûr !... Et c'est même relativement simple à mettre en œuvre !... Il te suffit pour cela de faire appel à l'instruction STEP que tu connais bien, et à donner au pas de progression une valeur qui soit différente de 1 !

O. — Oui, mais si je donne au pas de progression une valeur qui ne soit pas entière, comme 1/4 par exemple, que va-t-il se passer du point de vue affichage ?



P. — Il faudra tout simplement que la boucle soit parcourue quatre fois pour que l'engin avance d'un carreau, car tu as bien compris qu'il ne pouvait pas avancer d'un quart de carreau à chaque fois !

O. — Montre-moi un exemple !

P. — Ok !... On va choisir celui d'une voiture de course et on va successivement ralentir puis accélérer sa vitesse !... Cela te convient ?

O. — Tout à fait !

P. — Alors, voici pour commencer le programme d'un bolide qui évolue à une vitesse normale :

```
10 FOR C=0 TO 29
20 PRINT AT 10,C;"▟▙"
30 PRINT AT 11,C;"▜▛"
40 NEXT C
50 CLS
60 GOTO 10
```

O. — Je ne voudrais pas te décevoir, mais ta voiture de course avance d'un drôle de façon !

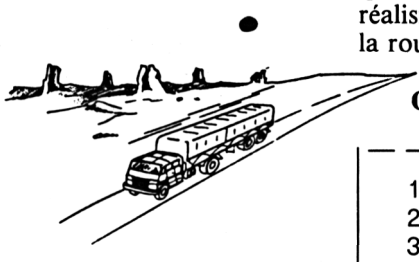
P. — Ça, c'est de ma faute !... Je t'explique que si tu dessines un engin qui se déplace horizontalement, tu dois te limiter à un carreau de hauteur, sinon il avance ligne par ligne, et je fais moi-même l'erreur !...

O. — Alors, que faire ?

P. — Changer d'engin !... Je ne vois pas d'autres solutions !... Que dirais-tu d'un camion semi-remorque ?

O. — Là, je ne comprends plus !... Il faut deux lignes pour dessiner un semi-remorque !... Alors, où est la différence ?

P. — La différence, c'est que la seconde ligne d'impression correspond aux roues et que, ce qui apparaissait comme un défaut pour la voiture de course ne fait qu'accentuer le réalisme du camion. On dirait que les roues bougent réellement, comme sur les cahots de la route !



O. — Astucieux !... Tu fais voir le programme ?

```
10 FOR C=0 TO 26
20 PRINT AT 10,C;"▟▙▟▙▟▙
```

O. — Si j'ai bien compris, le pas de progression doit être de Ø, 5 !... Pourquoi alors l'avoir écrit .5 ?

P. — J'ai voulu profiter de l'occasion pour te montrer que lorsque tu veux écrire un nombre décimal inférieur à 1, il est inutile d'écrire le Ø !... Ceci est typiquement anglo-saxon comme forme d'écriture !

O. — Ah bon !... On essaye une autre vitesse ?

P. — Bien sûr !... En voici une qui correspond au tiers de la vitesse normale :

```
10 FOR C=Ø TO 26 STEP 1/3
```

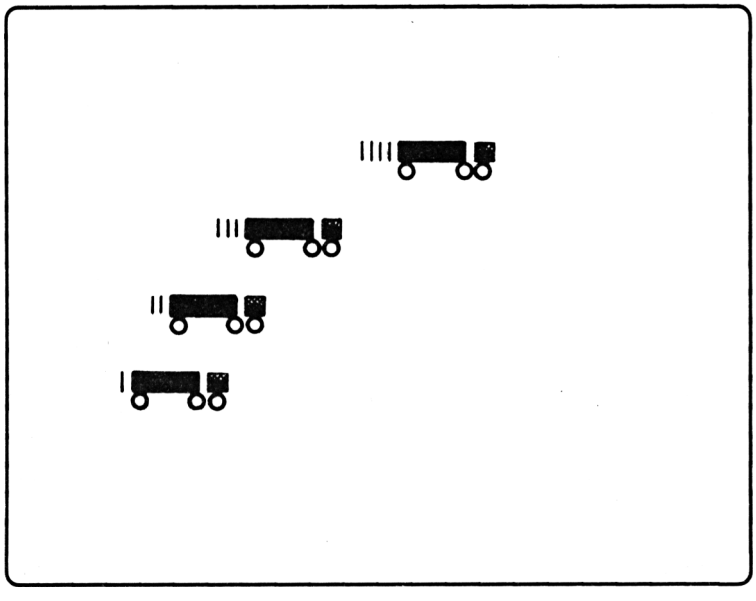
O. — Tiens, on peut écrire le pas sous la forme d'une fraction !

P. — Oui, et c'est bien pratique !... Surtout quand il s'agit d'un angle exprimé en radians !... On continue ?

O. — Ok, mais cette fois, c'est moi qui le fais !... Je te réserve d'ailleurs une surprise dont tu me diras des nouvelles :

```
 10 FOR C=Ø TO 26
 20 PRINT AT 4,C;" ████▒"
 30 PRINT AT 5,C;" O  OO"
 40 PRINT AT 8,C/2;" ████▒"
 50 PRINT AT 9,C/2;" O  OO"
 60 PRINT AT 12,C/3;" ████▒"
 70 PRINT AT 13,C/3;" O  OO"
 80 PRINT AT 16,C/4;" ████▒"
 90 PRINT AT 17,C/4;" O  OO"
100 NEXT C
110 CLS
120 GOTO 10
```

O. — Alors, qu'est-ce que tu penses de cela ?



P. — Mes compliments !... Tu viens de trouver tout seul l'autre méthode qui permet de modifier la vitesse d'un engin !

O. — Et tu noteras que j'ai dessiné quatre camions différents sur l'écran, chacun d'eux allant à une vitesse différente !

P. — J'ai noté !... J'ai noté également que tu ne pleurais ni ton temps, ni la place en mémoire.

O. — Pourquoi, mon programme est trop long ?

P. — Un peu !... Dis-toi bien que quand tu dessines quatre fois le même camion, c'est trois fois de trop !... Il y a toujours une solution qui permet d'éviter cela !

O. — Et comment ferais-tu dans le cas présent ?

P. — J'emploierais une seconde boucle du style FOR N = 1 TO 4, que je placerais à l'intérieur de l'autre, comme cela :

```
10 FOR C=0 TO 26
20 FOR N=1 TO 4
30 PRINT AT 4*N,C/N;"  ████ ▒▒"
40 PRINT AT 4*N+1,C/N;" O  OO"
50 NEXT N
60 NEXT C
70 CLS
80 GOTO 10
```

O. — Et ça marche exactement comme l'autre programme !... Chapeau !...

P. — N'est-ce pas ?... Remarque comment on détermine les lignes d'impression des camions et leur vitesse !... Pour les lignes, tu effectues le produit 4 *N, ce qui donne les lignes 4, 8, 12, et 16 pour les quatre camions !... Pour les vitesses, tu divises C par N, ce qui te permet d'obtenir les vitesses 1/1, 1/2, 1/3 et 1/4 !

O. — Ok, je m'en souviendrai à l'avenir !... Il reste toutefois un point qui me chiffonne !

P. — Dis vite !

O. — Lorsque le camion le plus rapide arrive sur le bord droit de l'écran, il s'efface pour réapparaître à gauche, mais en même temps qu'il s'efface, les autres camions s'effacent également !... Ce que j'aimerais, c'est qu'il soit le seul à s'effacer pour réapparaître, tandis que les autres continueraient leur chemin.

P. — Ce que tu me demandes là n'est pas facile à réaliser. Il faut rendre autonome chaque camion, et pour cela il faut employer des variables indicées !

Les variables indicées

O. — Ça y est, voilà les grands mots qui recommencent... Qu'est-ce que c'est que cette nouveauté ?

P. — Les variables indicées tirent leur nom du fait que chaque variable est suivie d'un indice !

O. — Tu veux parler d'un « truc » comme X_1, X_2, etc...

P. — Exactement !... Il s'agit d'une écriture qui permet d'associer une variable indépendante à chaque engin que tu souhaites voir évoluer avec des caractéristiques qui lui sont propres, tout en conservant une forme généralisante qui permet de simplifier les programmes.

O. — Alors là, je vais t'avouer que je n'ai rien compris !

P. — Je m'en doutais un peu que tu ne saisirais pas ce genre d'explications !... Prenons l'exemple des camions que tu souhaites rendre indépendants les uns des autres et appliquons-nous à résoudre ce problème avec une méthode classique. Ensuite, je te montrerai comment l'utilisation de variables indicées peut simplifier l'écriture d'un tel programme !

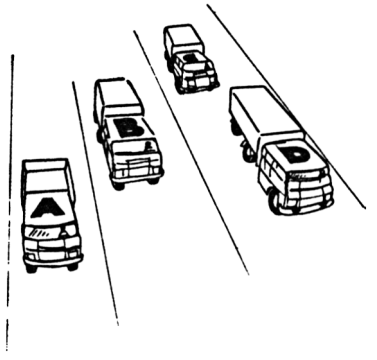
O. — Allons-y comme ça !

P. — Si tu veux rendre chaque camion indépendant, il faut associer une variable différente à chacun !... Qu'est-ce que tu choisis ?

O. — Quelque chose de tout simple !... La variable A pour le plus rapide, B pour le deuxième, C pour le troisième et D pour le plus lent !... Ça va comme ça ?

P. — Très bien !... Voici donc un « petit programme » qui va permettre à chaque camion d'être indépendant des autres :

```
10 LET A=0
20 LET B=0
30 LET C=0
40 LET D=0
50 LET A=A+1
60 LET B=B+1/2
70 LET C=C+1/3
80 LET D=D+1/4
130 PRINT AT 4,A;"
140 PRINT AT 5,A;" O  OO"
150 PRINT AT 8,B;"
160 PRINT AT 9,B;" O  OO"
170 PRINT AT 12,C;"
180 PRINT AT 13,C;" O  OO"
190 PRINT AT 16,D;"
200 PRINT AT 17,D;" O  OO"
210 GOTO 50
```



O. — Ton programme ressemble étrangement à celui que j'avais écrit précédemment et que tu trouvais trop long !

P. — Oui, n'est-ce pas ?... C'est justement pour te montrer comment l'emploi des variables indicées va permettre de le simplifier !... Mais avant, as-tu remarqué comment la boucle est écrite ?

O. — Tu n'as pas employé l'instruction FOR... TO... !... Pourquoi ?

P. — Parce qu'une telle instruction ne peut faire changer de valeur qu'une seule variable à la fois !... A partir du moment où tu en as plusieurs, tu dois changer de méthode !

O. — Ah bon !

P. — Tu vas remarquer comment les deux méthodes se ressemblent !... D'abord, les lignes 10 à 40 fixent la valeur de départ de chaque variable. Ensuite, ce sont les lignes 50 à 80 qui déterminent le pas de progression de chacune des variables. Enfin, pour terminer, tu vas trouver, après la longue série des instructions PRINT, la ligne 210 qui te renvoie aux lignes 50 à 80.

O. — Cette dernière ligne équivaut à l'instruction NEXT, si j'ai bien compris !...

P. — Exact !

O. — Alors, dans ces conditions je ne vois pas les lignes d'instructions qui correspondent aux valeurs de fin de boucle !

P. — J'ai gardé cela pour la bonne boucle, pardon, la bonne bouche, car ce n'est pas évident !... Voici déjà ce qui va te permettre d'effacer et faire réapparaître le premier camion sans toucher aux autres :

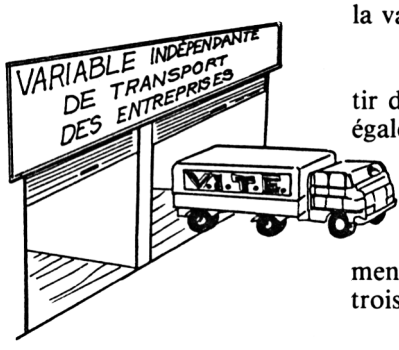
```
 90 IF A>=26.5 THEN GOSUB 220
220 PRINT AT 4,A;"      "
230 PRINT AT 5,A;"      "
240 LET A=0
250 RETURN
```

O. — Je vois que tu fais appel à un sous-programme dès que la valeur de A est supérieure ou égale à 26,5 !... Quelle drôle de valeur !

P. — Je vais t'expliquer !... D'abord le sous-programme !... As-tu remarqué qu'il servait à la fois à effacer le premier camion et à remettre à zéro sa variable A de progression ?

O. — Oui !... Il est évident qu'on ne peut pas employer l'instruction CLS pour effacer un seul camion !

P. — Quant à la valeur bizarre de 26,5, elle provient des règles d'arrondissement employé pour l'affichage !.. Il faut que tu saches que toute la valeur décimale jusqu'à .49999999 est arrondie à la valeur inférieure et qu'à partir de .500000000, elle est arrondie à la valeur supérieure.



O. — Ah, je comprends !... Comme il faut interdire l'impression d'un camion à partir de la 27e colonne, sinon il déborderait de l'écran, il faut donc détecter toute valeur égale ou supérieur à 26,5 puisque, avec la règle d'arrondissement, cela donnerait 27 !

P. — Exactement !... Maintenant, dis-moi ce que tu penses de ce programme !

O. — J'en pense que, pour le moment, cela marche pour un camion !... Effectivement, quand il arrive en bord d'écran, il s'efface pour réapparaître à droite, tandis que les trois autres continuent leur course comme si de rien n'était !

P. — Ceci, c'est parce que tu contrôles le déplacement de chaque camion à l'aide d'une variable indépendante !... Ecris donc maintenant la ligne 100 et le sous-programme 260 pour le deuxième camion !

O. — J'étais déjà en train de le faire !... Les voici :

```
100 IF B>=26.5 THEN GOSUB 260
260 PRINT AT 8,B;"      "
270 PRINT AT 9,B;"      "
280 LET B=0
290 RETURN
```

O. — Zut, ça recommence !... J'ai déjà une saturation de la mémoire d'affichage !

P. — Je m'en doutais un peu qu'elle n'allait pas tarder à se produire, et c'est pourquoi j'ai voulu t'éviter d'écrire pour rien la fin du programme pour les troisième et quatrième camions !

O. — Merci !... En tout cas, si tu voulais me faire comprendre que ce programme est trop long, tu as gagné !

P. — A mon tour de te remercier de cette perche que tu veux bien me tendre pour te présenter l'outil qui va te permettre de le simplifier, j'ai nommé les variables indicées.

O. — Allons-y donc, puisqu'il n'y a pas moyen d'y échapper !

P. — Pour commencer, tu vas remplacer les quatre variables que tu as employées, c'est-à-dire A, B, C et D, par C(1), C(2), C(3) et C(4).

O. — Si je comprends bien, ce sont les indices 1, 2, 3 et 4 qui vont permettre de distinguer chaque variable l'une de l'autre !

P. — C'est tout à fait cela !

O. — Est-ce que je suis tenu d'utiliser obligatoirement la lettre C ?

P. — Non, absolument pas !... Par contre, les indices doivent obligatoirement commencer par le chiffre 1 et se suivre dans l'ordre !

O. — Bien !... Et que dois-je faire après ?

P. — Après, tu remplaces les lignes 1Ø à 4Ø dans lesquelles tu initialises à zéro tes quatre variables par la ligne unique suivante :

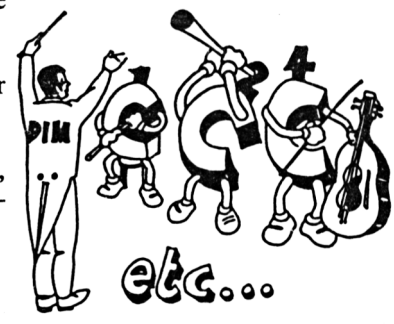
```
10 DIM C(4)
```

O. — Tiens, une nouvelle instruction !

P. — Eh oui !... L'instruction DIM te permet de « dimensionner » un tableau, c'est-à-dire de définir le nombre d'indices que tu veux affecter à une variable.

O. — Ce qui, dans notre cas, correspond à quatre !

P. — C'est ça !... De surcroît, cette ligne d'instruction t'initialise chaque variable indicée, comme le faisaient les lignes 1Ø à 4Ø !



O. — C'est effectivement plus rapide !... Y-a-t'il un maximum à ne pas dépasser dans le nombre d'indices qu'on peut attribuer à une variable ?

P. — Très bonne question !... Théoriquement, il n'y a pas de limite !... En pratique, c'est la taille de la mémoire qui va en imposer une, dans la mesure où chaque indice immobilise l'équivalent d'une ligne d'instruction !

O. — Et après avoir dimensionné ma variable, qu'est-ce que je fais ?

P. — Tu vas généraliser l'écriture de tes indices, en remplaçant les chiffres par une lettre unique, tel que N par exemple. Ainsi, au lieu d'écrire C(1) à la ligne 5Ø, C(2) à la ligne 6Ø, C(3) à la ligne 7Ø et C(4) à la ligne 8Ø, en remplacement des variables A, B, C et D, tu ne vas écrire qu'une seule fois C(N), comme ceci :

```
50 LET C(N)=C(N)+1/N
```

O. — Ah oui, je crois comprendre !... Comme la lettre N peut prendre les valeurs de 1 à 4, cette ligne que tu viens d'écrire équivaut à la ligne 5Ø précédente si N vaut 1, à la ligne 6Ø si N vaut 2, à la ligne 7Ø si N vaut 3 et à la ligne 8Ø si N vaut 4. C'est ça ?

P. — Tout à fait !... Seulement, pour obtenir cela, il faut placer cette instruction dans une boucle où tu fais varier N de 1 à 4, sinon N n'est pas défini !... Comme tu as la place, tu n'as qu'à la numéroter 2Ø :

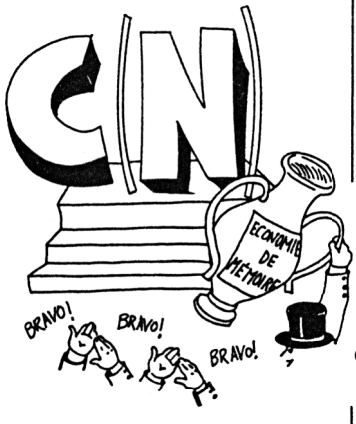
```
20 FOR N=1 TO 4
```

O. — Astucieux, tout ça !... Si je comprends bien, je vais pouvoir remplacer, maintenant que N est défini, chaque groupe de quatre lignes d'instructions par une seule, dans laquelle les variables figurent sous une écriture unique : C(N) !... Quelle économie !

P. — Je te crois !... Tu divises par quatre la quantité de mémoire nécessaire !

O. — Bien, alors je me lance !... Voici ce que donne cette nouvelle écriture :

```
90 IF C(N)>=26.5 THEN GOTO 220
130 PRINT AT 4*N,C(N);"  "
140 PRINT AT 4*N+1,C(N);" O  OO
"
220 PRINT AT 4*N,C(N);"      "
230 PRINT AT 4*N+1,C(N);"      "
240 LET C(N)=0
250 RETURN
```

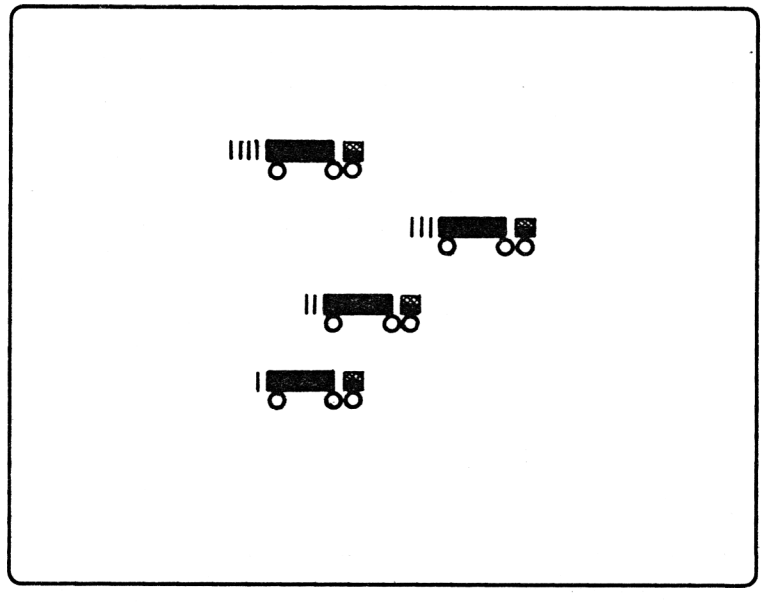


P. — Bravo !... Tu essayes pour voir ce que cela donne ?

O. — Tiens, ça ne marche pas !... Ah, j'ai trouvé : il manque les lignes de fermeture de la boucle :

```
200 NEXT N
210 GOTO 20
```

P. — Et voilà le travail !... Alors, ton avis sur les variables indicées ?

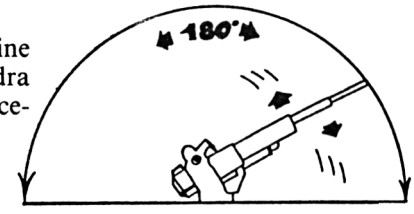


O. — C'est super !... Et quelle simplification de l'écriture des programmes cela permet !

Les déplacements en oblique

P. — Maintenant que tu sais comment faire évoluer en même temps plusieurs engins sur l'écran, indépendamment les uns des autres, il te reste à voir comment peuvent s'effectuer des déplacements en oblique !

O. — Boaff, c'est pas très passionnant ce « truc »-là !... A quoi donc est-ce que cela peut servir ?



P. — Tout simplement à mettre en place tes futurs programmes de tir !... J'imagine que si tu cherches à protéger une station de l'espace d'une pluie d'astéroïdes, il te faudra pouvoir orienter le tir de ton canon laser !... C'est à cela que vont te servir les déplacements en oblique !

O. — Dans ces conditions, cela change tout !... Par quoi on commence ?

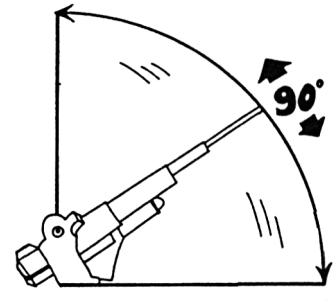
P. — Plutôt que de te dire par quel type de déplacement nous allons commencer, j'aimerais que tu réfléchisses un peu aux divers emplacements sur l'écran où tu serais susceptible de vouloir installer un canon laser !

O. — Tout d'abord, en plein milieu de l'écran !... Il faudrait qu'il puisse s'orienter dans toutes les directions puisque les astéroïdes sont susceptibles d'arriver de n'importe quel point de l'espace !

P. — Très bien !... Et ensuite ?

O. — Ensuite, je me vois bien en train de défendre une base lunaire à partir d'un canon au sol, placé au milieu de la dernière ligne de l'écran !

P. — Dans ce cas, tu n'aurais plus à l'orienter que sur 180 degrés, ce qui représente déjà une simplification par rapport à l'emplacement précédent où là, il te fallait l'orienter sur 360 degrés !...



O. — Quant au dernier emplacement possible, j'imagine que cela peut être l'un des coins inférieurs de l'écran !

P. — Exact !... Et, cette fois, son orientation ne dépasse pas 90 degrés !... C'est d'ailleurs par ce dernier emplacement que nous allons débuter car, des trois, c'est le plus simple à mettre en œuvre !

O. — Ok !... Et avec quelle instruction on fait ça ?... PRINT ou PLOT ?

P. — PLOT ce coup-ci !... On obtiendra ainsi un graphisme plus fin !

O. — Très bien !... Si tu veux, je vais commencer par le déplacement à 45 degrés !

P. — D'accord, mais à partir de quel coin démarres-tu ?

O. — Le gauche !... Voici d'ailleurs le programme :

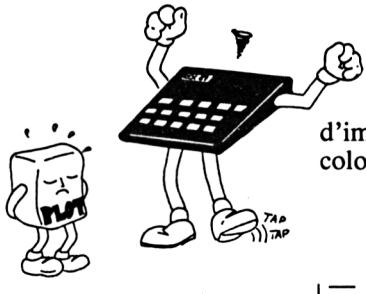
```
10 FOR X=0 TO 44
20 IF X>0 THEN UNPLOT X-1,X-1
30 IF X<44 THEN PLOT X,X
40 NEXT X
50 GOTO 10
```

P. — Je vois que tu es passé maître dans l'art d'écrire des programmes de déplacement. Il marche du premier coup !

O. — Puisqu'il te convient, je vais essayer de faire aussi bien avec celui du déplacement à 30 degrés :

```
10 FOR X=0 TO 44
20 IF X>0 THEN UNPLOT 2*X-2,X-
1
30 IF X<44 THEN PLOT 2*X,X
40 NEXT X
50 GOTO 10
```

P. — Et alors ?



O. — Alors, ça ne marche pas !... J'obtiens un message d'erreur B/30 !

P. — Et cela t'étonne ?... Regarde un peu ta ligne 30 : tu demandes à l'ordinateur d'imprimer un demi-carré dans la demi-colonne 84 lorsque X vaut 42. Or, des demi-colonnes, il n'y en a que 64 !

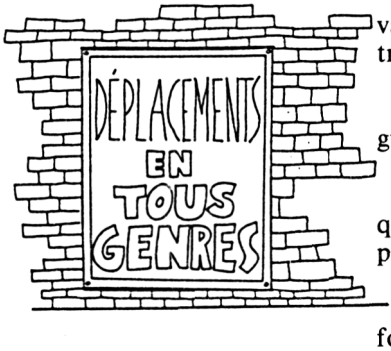
O. — Oh, l'erreur !... Ne dis rien, je rectifie cela tout de suite sur les lignes 10 et 30 :

```
10 FOR X=0 TO 32
30 IF X<32 THEN PLOT 2*X,X
```

P. — Voilà, c'est mieux !... Il ne te reste plus qu'à écrire celui à 60 degrés et tu auras ainsi réalisé les trois déplacements obliques qu'il est possible de concevoir à partir de ce coin de l'écran !

O. — C'est comme si c'était fait !... Et voilà le travail :

```
10 FOR X=0 TO 22
20 IF X>0 THEN UNPLOT X-1,2*X-
2
30 IF X<22 THEN PLOT X,2*X
40 NEXT X
50 GOTO 10
```



P. — Bravo !... Maintenant que tu as écrit correctement ces trois programmes, je vais te demander d'essayer d'élaborer un programme universel capable de réaliser ces trois déplacements en oblique !

O. — Hou là là, c'est pas simple ce que tu me demandes là !... Il va falloir que tu me guides un peu !

P. — D'accord !... Alors, pour t'aider, je vais commencer par te faire remarquer que ton déplacement à 45 degrés est deux fois plus lent que les deux autres !... Ce n'est pas normal !

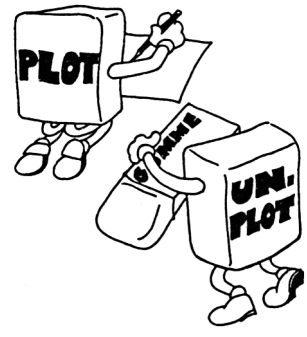
O. — Ça, je sais d'où ça vient !... C'est uniquement dû au fait que j'imprime deux fois plus de demi-carrés que dans les deux autres !... Je vais t'arranger cela en deux coups de cuillère à pot !... Il suffit de modifier les lignes 10, 20 et 30 comme suit :

```
10 FOR X=0 TO 22
20 IF X>0 THEN UNPLOT 2*X-2,2*
X-2
30 IF X<22 THEN PLOT 2*X,2*X
```

P. — Bien !... Ensuite, tu vas remplacer les chiffres que tu utilises dans les coordonnées des instructions PLOT et UNPLOT par deux variables I et J, la variable I étant associée à la coordonnée des demi-colonnes et J à celle des demi-lignes !... Un conseil avant que tu ne te lances à corps perdu dans cette modification : n'oublie pas de définir la valeur de I et de J en début de chaque programme !

O. — Ok !... Je commence donc par le déplacement à 45 degrés :

```
10 LET I=2
20 LET J=2
30 FOR X=0 TO 22
40 IF X>0 THEN UNPLOT I*X-I,J*
X-J
50 IF X<22 THEN PLOT I*X,J*X
60 NEXT X
70 GOTO 30
```



P. — Parfait !... Tu continues par lequel ?

O. — Par celui à 30 degrés !... Voilà ce que cela donne :

```
10 LET I=2
20 LET J=1
30 FOR X=0 TO 32
40 IF X>0 THEN UNPLOT I*X-I,J*
X-J
50 IF X<32 THEN PLOT I*X,J*X
60 NEXT X
70 GOTO 30
```

P. — Tu ne remarques rien ?

O. — Non !... Pourquoi ?

P. — Alors, écris le dernier programme !... Là, je pense que tu devrais t'apercevoir de quelque chose de particulier !

O. — Ok !... Voici donc le programme de déplacement à 60 degrés, nouvelle formule :

```
10 LET I=1
20 LET J=2
30 FOR X=0 TO 22
40 IF X>0 THEN UNPLOT I*X-I,J*
X-J
50 IF X<22 THEN PLOT I*X,J*X
60 NEXT X
70 GOTO 30
```

P. — Et tu ne remarques toujours rien ?

O. — Si justement !... Je m'en suis rendu compte en écrivant la ligne 4Ø !... Dans les trois programmes, cette ligne est la même !

P. — Tu as trouvé !... Tu peux même dire qu'il en est de même pour la ligne 5Ø, à l'exception de la valeur limite qui est égale à 22 dans les deux premiers programmes et à 32 dans le troisième !

O. — Oui, c'est vrai ça !... Je me demande comment on pourrait faire pour les rendre identiques toutes les trois ?

P. — Je ne dirais pas que cela est évident, mais tu vas trouver la solution toute simple dès que je t'aurai fait remarquer que la valeur limite de 32 n'apparaît que dans le programme où J = 1.

O. — Ça y est, j'ai compris !... Il suffit de calculer cette valeur limite avec l'opération :

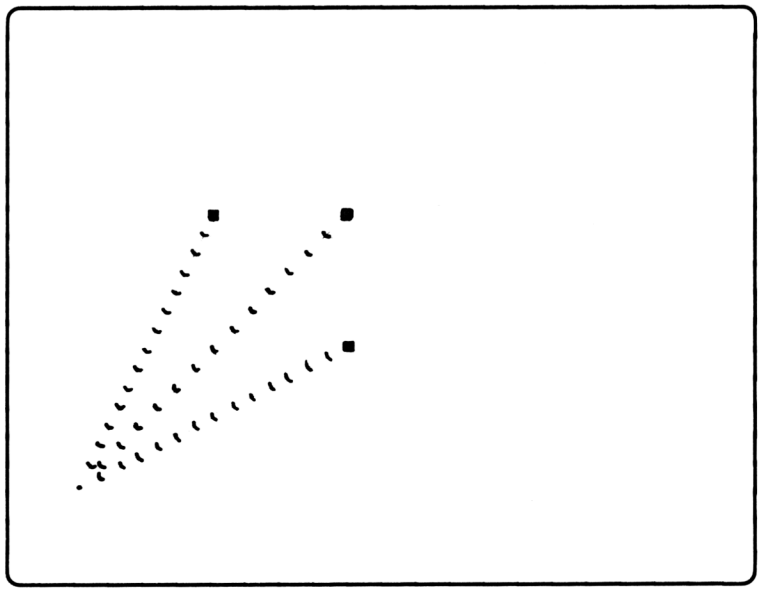
```
25 LET M=42-J*1Ø
```

P. — Exact !... Ainsi quand J vaut 1, M est égal à 42 – 1Ø = 32 et quand J vaut 2, M est égal à 42 – 2Ø = 22 !

O. — Il ne me reste plus qu'à reporter cette valeur limite M dans les lignes 3Ø et 5Ø, et on aura ainsi obtenu le programme universel qu'on cherchait :

```
3Ø FOR X=Ø TO M
5Ø IF X<M THEN PLOT I*X,J*X
```

P. — Eh oui !... Il suffit maintenant de changer les valeurs de I ou de J pour changer l'angle de la trajectoire oblique !

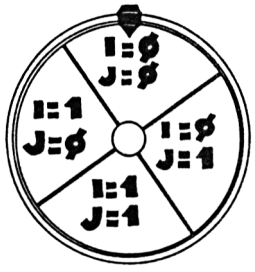


O. — Et si on s'amusait à faire un programme qui change tout seul l'angle de cette trajectoire ?

P. — Bonne idée !... Remplace donc tes lignes 1Ø et 2Ø dans lesquelles tu définis I et J, par celles-ci :

```
10 LET I=INT (RND*2)+1
15 LET J=INT (RND*2)+1
20 IF I=1 AND J=1 THEN GOTO 10
```

O. — C'est astucieux cette façon de faire !... Tu calcules au hasard les valeurs de I et de J qui peuvent valoir 1 ou 2. Mais à quoi sert la ligne 2Ø ?



P. — A empêcher que I et J ne puissent avoir tous les deux la valeur 1 en même temps, cette solution ne correspondant à aucun des trois déplacements que nous venons d'étudier !

O. — C'est vrai !... Bon, il ne reste plus qu'à modifier également la ligne 7Ø pour que ce programme marche de façon automatique :

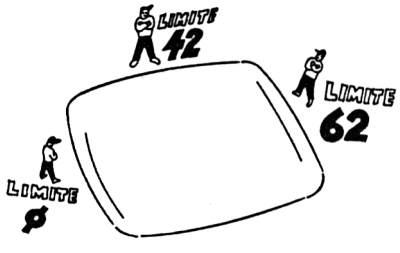
```
70 GOTO 10
```

P. — Eh bien, voilà une bonne chose de faite !... Maintenant, on va s'occuper des déplacements en oblique sur 180 degrés !

O. — Je suppose qu'il suffit pour cela d'employer le même programme que précédemment, en changeant seulement les valeurs des deux variables I et J ?

P. — Tu supposes juste !... Modifie donc ton programme précédent en conséquence !

O. — Voilà, c'est fait !... Il a suffit que j'étende les valeurs de I de +2 à −2. Quant à J, ses valeurs demeurent inchangées !




```
10 LET I=INT (RND*5)-2
15 LET J=INT (RND*2)+1
20 IF I=1 AND J=1 OR I=Ø AND J
=1 OR I=-1 AND J=1 THEN GOTO 10
```

P. — Je vois que tu as conservé le programme de changement automatique de trajectoire !

O. — Oui, il était trop chouette !... Par contre, il a fallu que j'interdise trois combinaisons parmi les dix possibles car il n'y a que sept trajectoires différentes.

P. — Bien !... Et comment vas-tu t'y prendre pour calculer les valeurs limites ?

O. — Alors là, je ne sais pas trop !... Maintenant, il y en a trois, une par bordure d'écran !

P. — Dans ces conditions, le calcul de tout à l'heure n'est plus valable !... Il te faut trouver une solution générale qui marche dans tous les cas !

O. — Je viens d'avoir une idée !... Il suffit que je détecte la sortie du projectile sur l'un des trois côtés de l'écran en vérifiant en permanence la valeur des coordonnées de l'instruction PLOT !

P. — Bravo !... Ainsi, si I*X+32 devient égal à Ø ou à 62, c'est que le projectile est arrivé sur le bord gauche ou droit de l'écran. Par contre, si c'est J*X qui est égal à 42, c'est qu'il est arrivé en haut de l'écran !

O. — J'ai deux questions à te poser !... D'abord, pourquoi avoir rajouté 32 à I*X ?

P. — Parce que ce nombre correspond à la position de l'origine de la trajectoire oblique qu'on a décidé de placer au milieu de la ligne inférieure de l'écran !

O. — C'est vrai !... Et pourquoi employer les valeurs 42 et 62 puisque l'écran comporte 43 demi-lignes et 63 demi-colonnes ?

P. — Tout simplement parce que le projectile progresse de deux en deux petits carrés et que, partant d'une origine aux coordonnées paires (32, Ø), il fallait fixer des limites paires !

O. — Ça, je n'y aurais pas pensé !... Bien, alors je réécris ma ligne 25 :

```
25 IF I*X+32=Ø OR J*X=42 OR I*
X+32=62 THEN LET M=I*X+32 OR J*X
```

P. — Et tu crois que cela va marcher ?... Je vois deux erreurs fondamentales dans cette ligne d'instruction !

O. — Ah bon !... Lesquelles ?

P. — Tout d'abord, tu ne peux donner deux valeurs différentes à une même variable après l'instruction LET !... Ensuite, tu ne peux détecter à l'extérieur d'une boucle des valeurs qui dépendent de la variable de boucle !

O. — Tiens, c'est vrai ça !... Mais alors comment faire ?

P. — Il va te falloir écrire un programme de progression en oblique qui n'emploie pas les instructions FOR...TO et NEXT !

O. — C'est possible ça ?

P. — Bien sûr !... Je vais d'ailleurs te l'écrire en le commentant au fur et à mesure !... D'abord, il faut fixer la valeur initiale de la variable X, comme cela :

```
30 LET X=Ø
```

P. — Ensuite, tu écris les lignes d'effacement et d'impression, en plaçant entre les deux la ligne de sortie de boucle :

```
40 IF X>Ø THEN UNPLOT I*X-I+32
,J*X-J
50 IF I*X+32=Ø OR J*X=42 OR I*
X+32=62 THEN GOTO 10Ø
60 PLOT I*X+32,J*X
```

O. — Ta ligne de sortie de boucle ressemble étrangement à celle que j'avais tenté d'écrire !

P. — N'est-ce pas ?... Seulement, dans ce cas, je m'en sers pour réinitialiser la boucle, comme si on avait atteint la valeur maximale avec l'instruction FOR...TO.

O. — Je remarque à ta ligne 60 que tu as supprimé la condition IF X <M !... Pourquoi ?

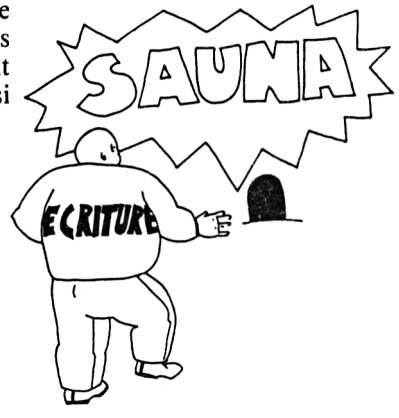
P. — Parce que cette condition figure à la ligne 50 et que si elle est réalisée, la ligne 60 n'est jamais atteinte puisque le déroulement du programme saute à la ligne 10.

O. — C'est vrai !

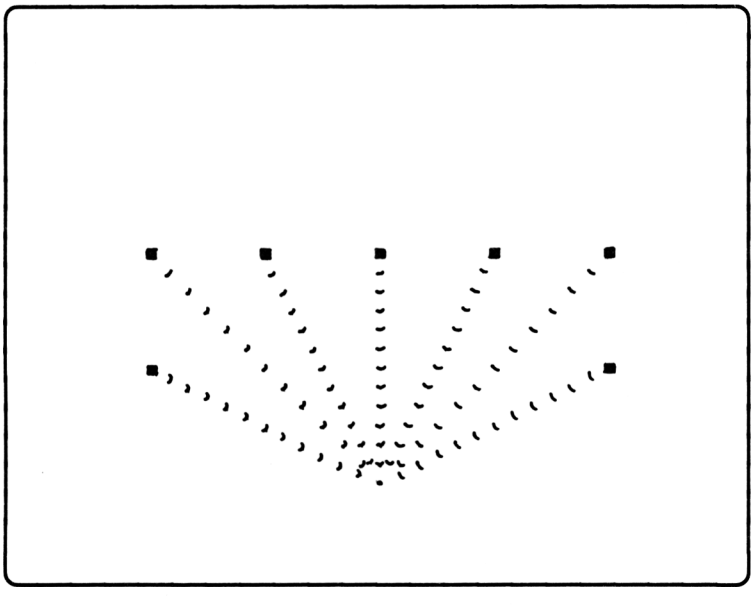
P. — Je termine donc ce programme avec deux lignes d'instruction qui remplacent celle de l'intruction NEXT :

```
70 LET X=X+1
80 GOTO 40
```

O. — Et ça marche !... Toutefois, si tu me permets une petite critique, je trouve que ton écriture est un peu lourde !... Si tu veux bien, je vais poser deux variables que je vais appeler A et B et qui vont remplacer respectivement les I*X+32 et J*X qui se répètent un peu trop souvent dans tes lignes 40, 50 et 60 !... Voilà ce que donne le programme ainsi réécrit :




```
30 LET X=0
32 LET A=I*X+32
35 LET B=J*X
40 IF X>0 THEN UNPLOT A-I,B-J
50 IF A=0 OR B=42 OR A=62 THEN
GOTO 10
60 PLOT A,B
70 LET X=X+1
80 GOTO 32
```

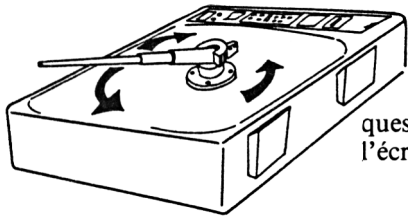


P. — Mes compliments pour cette heureuse initiative !... D'abord, cela simplifie

l'écriture, et ensuite, cela accélère la vitesse du déplacement en oblique puisque chaque opération I*X+32 et J*X n'est effectuée qu'une seule fois !

O. — Par contre ce qui est couillon, c'est qu'au bout d'un moment on obtient une saturation de la mémoire !

P. — A ce moment-là, supprime la ligne 20 !... Tu vas voir qu'elle n'est pas foncièrement indispensable !



O. — Exact !... Cela améliore un peu la situation !

P. — Bien !... Il ne nous reste plus qu'à terminer cette étude sur les trajectoires obliques par le morceau de bravoure que représente le tir sur 360 degrés à partir du centre de l'écran !

O. — Comme tu dis !... Et je parie que tu comptes sur moi pour faire ça ?

P. — Comment as-tu deviné ?... En fait, le programme qui permet d'obtenir cela ne diffère guère du programme précédent élaboré ensemble que par un ou deux points de détail !

O. — Ok !... Alors, je me jette à l'eau !... Qu'est-ce que tu penses de cela :

```
 10 LET I=INT (RND*5)-2
 20 LET J=INT (RND*5)-2
 30 LET X=0
 40 LET A=I*X+32
 50 LET B=J*X+22
 60 IF X>0 THEN UNPLOT A-I,B-J
 70 IF A=0 OR B=42 OR A=62 OR B
=0 THEN GOTO 10
 80 PLOT A,B
 90 LET X=X+1
100 GOTO 40
```

P. — Je vois que tu as supprimé la ligne qui permettait d'éliminer les combinaisons interdites de I et de J !

O. — Tu penses !... J'ai calculé que sur 25 combinaisons il fallait que j'en supprime 9 puisqu'il n'existe que 16 trajectoires différentes !

P. — Dans ces conditions, que dirais-tu de remplacer ce choix au hasard par un petit programme qui fait tourner tout seul le tir ?

O. — Génial comme idée !... Et comment peut-on obtenir cela ?

P. — En écrivant les quelques lignes suivantes :

```
  5 LET I=0
  7 LET J=2
 10 IF J=2 AND I<2 THEN LET I=I
+1
 15 IF I=-2 AND J<2 THEN LET J=
J+1
 20 IF J=-2 AND I>-2 THEN LET I
=I-1
 25 IF I=2 AND J>-2 THEN LET J=
J-1
```

O. — Je crois que je comprends !... J'avais remarqué que, pour tourner, il fallait faire varier l'une des deux variables I ou J entre +2 et −2 ou −2 et +2, tout en maintenant l'autre à +2 ou à −2 !

P. — C'est tout à fait cela !... C'est pourquoi je t'ai écrit quatre lignes différentes, chacune d'entre-elles assurant une rotation de 90 degrés.

O. — Très astucieux tout cela !... Par contre, ce qui l'est moins, c'est que j'ai à nouveau une saturation de la mémoire !

P. — Alors, on applique les méthodes qui permettent d'économiser la mémoire !... Comme les chiffres Ø, 1 et 2 se répètent souvent, je te propose de les remplacer par des lettres !

O. — Ok !... Voici donc les lignes à rajouter et à modifier :

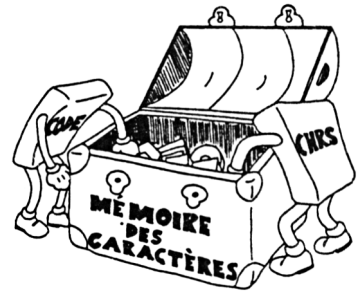
```
  1 LET Z=PI-PI
  2 LET U=PI/PI
  3 LET D=U+U
  5 LET I=Z
  7 LET J=D
 10 IF J=D AND I<D THEN LET I=I
+U
 15 IF I=-D AND J<D THEN LET J=
J+U
 20 IF J=-D AND I>-D THEN LET I
=I-U
 25 IF I=D AND J>-D THEN LET J=
J-U
 30 LET X=Z
 60 IF X>Z THEN UNPLOT A-I,B-J
 70 IF A=Z OR B=42 OR A=62 OR B
=Z THEN GOTO 10
 90 LET X=X+U
```

P. — Et alors ?

O. — C'est mieux, bien sûr, mais ça sature toujours !... Il n'y a pas moyen de remplacer également les nombres 1Ø, 22, 32, 42 et 62 par des lettres ?

P. — Si, mais pas par des lettres !... Il te faut employer pour cela l'intruction CODE !

O. — Tiens, une nouvelle instruction !... A quoi sert-elle ?



P. — L'instruction CODE, suivie de l'un des 256 caractères du micro-ordinateur placé entre guillemets, délivre le numéro de code de ce caractère !... Tu trouveras la liste de tous les caractères, ainsi que leur numéro de code à la page 181 de ton manuel d'utilisation !

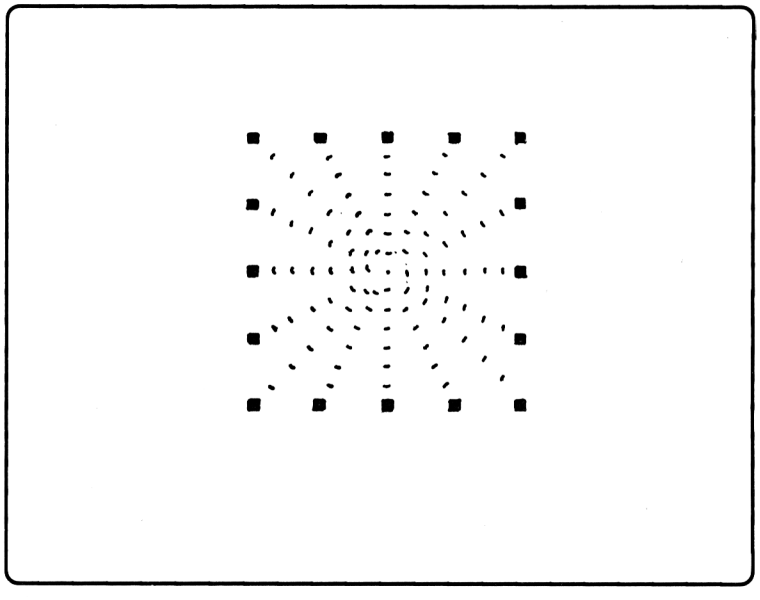
O. — C'est exactement l'inverse de l'instruction CHR $!

P. — Tout à fait !... Il te suffit maintenant de remplacer chaque nombre par cette instruction et tu économiseras encore un peu de place dans ta mémoire !... Mais je t'avertis : le gain de place est très faible !

O. — Ok, j'essaye toujours !... Voici ce que cela donne :

```
40 LET A=I*X+CODE "4"
50 LET B=J*X+CODE "-"
70 IF A=Z OR B=CODE "E" OR A=C
ODE "Y" OR B=Z THEN GOTO CODE "
"
100 GOTO CODE "C"
```

P. — Bravo !... Et que devient la saturation ?



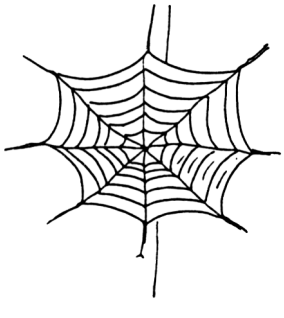
O. — Ça va un peu mieux, mais c'est pas encore ça !

P. — Alors je ne vois plus qu'une seule solution : limiter les trajectoires vers le côté droit de l'écran en diminuant un peu la valeur de A = 62 dans la ligne d'instruction 70.

O. — C'est bon !... Ça marche avec A = 52, ce qui correspond à CODE « O ».

P. — Parfait !... Maintenant que ça marche, je vais te suggérer de suprimer la ligne 60 qui comporte l'instruction UNPLOT afin que tu puisses conserver à l'écran l'ensemble des 16 trajectoires !

O. — Bonne idée !... Ça y est, c'est parti !... C'est superbe, on dirait une sorte de toile d'araignée !



P. — Tu ne remarques rien ?

O. — Si, il manque une trajectoire !... C'en est une qui part à 45 degrés !...

P. — Malheureusement, je ne connais pas de solution qui permette de les obtenir toutes les seize !... Si tu en découvres une, tiens-moi au courant !

O. — Ok !... Je chercherai quand j'aurai un moment !

P. — En attendant que tu trouves, il faut que je te parle de la trajectoire balistique !

La trajectoire balistique

O. — La trajectoire balistique ?... Qu'est-ce que c'est que cela ?

P. — C'est la trajectoire d'un corps lancé en l'air et qui retombe sur terre grâce à l'action de la pesanteur !

O. — Comme quand je lance une balle de tennis le plus loin possible ?

P. — Tout à fait !... La trajectoire que décrit ta balle de tennis dans l'air est ce qui s'appelle une trajectoire balistique !

O. — Ah bon !... Je faisais de la trajectoire balistique sans le savoir !...

P. — Comme Monsieur Jourdain faisait de la prose !... Mais trève de bavardage, je voudrais que tu réfléchisses aux paramètres qui déterminent une trajectoire balistique !

O. — Les paramètres ?... C'est quoi un paramètre ?

P. — C'est ce qui caractérise une trajectoire parmi toutes les autres, comme l'angle de tir par exemple !

O. — Ou la vitesse de départ de la balle ?

P. — C'est ça !... Il y a même la vitesse du vent et sa direction qui interviennent !

O. — Hou la la !... Ne me dis pas qu'il va falloir tenir compte de tout cela ?

P. — Si !... Mais rassure-toi, on ne va pas tout faire en même temps !... On va d'ailleurs commencer par étudier l'influence de la vitesse de départ de la balle, liée à l'action de la pesanteur ?

O. — Et l'angle de tir ?

P. — On le laissera fixe, égal à 45 degrés !...

O. — Ok !... Alors, à toi l'honneur !

P. — On va donc appeler X et Y les coordonnées horizontales et verticales de déplacement de la balle. Voici pour commencer leur équation mathématique :

$$x = V_0 t$$
$$y = V_0 t - \frac{1}{2} g t^2$$

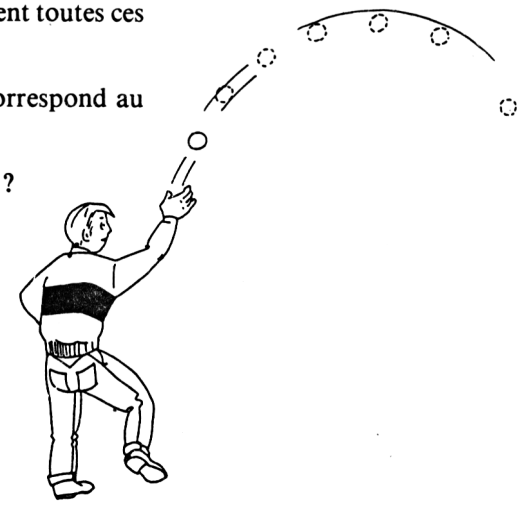
O. — Moi, je veux bien, mais il faudrait m'expliquer à quoi correspondent toutes ces lettres !

P. — Bien sûr !... V_0 est la vitesse de départ de la balle. La lettre t correspond au temps qui s'écoule. Quant à la lettre g, il s'agit de la pesanteur !

O. — Ah bon !... Et comment tu fais pour écrire tout cela en BASIC ?

P. — Tu procèdes ainsi :

```
10 LET V=1
20 LET G=.02
30 LET T=0
40 LET X=V*T
50 LET Y=V*T-G*T*T
60 PLOT X,Y
70 LET T=T+1
80 GOTO 40
```



O. — Ah oui !... Je retrouve aux lignes 4Ø et 5Ø les deux équations de la trajectoire balistique !... Comment tu as fait pour déterminer les valeurs de V et de G ?

P. — Par tâtonnements successifs !... J'ai cherché les valeurs qui permettaient d'obtenir la plus belle trajectoire !

O. — Je ne voudrais pas critiquer, mais elle n'est pas géniale ta trajectoire !... Elle n'est pas suffisamment arrondie pour commencer et ensuite, elle repart vers le haut de l'écran après avoir touché le sol !

P. — On va arranger cela tout de suite !... Voilà déjà de quoi arrondir la courbe !... Tu remarqueras que j'ai espacé les valeurs de T de trois en trois :

```
70 LET T=T+3
```

O. — C'est effectivement nettement mieux !

P. — Bien !... Pour le rebond, qui n'est rien d'autre que la partie souterraine de la trajectoire, il suffit d'interdire son affichage dès que Y devient inférieur à zéro :

```
55 IF Y<Ø THEN STOP
```

O. — Tiens, voilà une nouveauté !... Heureusement, elle n'est pas difficile à comprendre !

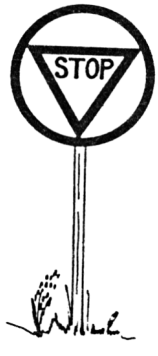
P. — N'est-ce pas ?... L'instruction STOP, comme son nom l'indique, est une instruction d'arrêt obligatoire. Elle arrête le déroulement d'un programme à l'endroit où elle est placée !

O. — Ok, c'est enregistré !... Maintenant, comment je fais si je veux changer de vitesse ?

P. — Tu interviens au niveau de la ligne 1Ø et tu changes la valeur de V !

O. — Et qu'est-ce que je mets comme valeur ?

P. — Essaye avec .5 ou 2 !... Tu verras bien ce qui se passe !



O. — C'est parti avec .5 !... Oh, j'obtiens une petite trajectoire ridiculement courte !... Par contre, avec 2, je sors de l'écran et j'obtiens un message d'erreur B/6Ø.

P. — Il te faut donc contrôler la sortie de ta balle de l'écran en rajoutant à ta ligne 55 :

```
55 IF Y<Ø OR X>63 THEN STOP
```

O. — Maintenant c'est bon !... J'obtiens un message 9/55 !

P. — Bravo !... Puisque tu as bien saisi le fonctionnement de ce programme, on va compliquer un peu. Tu as remarqué que ta variable était le temps et non une coordonnée horizontale ou verticale comme c'est le cas habituellement ?

O. — Oui !... Et alors ?

P. — Cela se traduit donc par des valeurs décimales pour X et pour Y, ce qui ne faci-

litera pas par la suite la détection de la position de la balle si, par exemple, tu veux savoir si elle est entrée dans un trou !

O. — Ça c'est sûr !

P. — Alors, voici une petite modification que l'on peut faire au programme précédent et qui permet de progresser par valeurs entières de X :

```
30 LET X=0
40 LET T=X/V
50 LET Y=X-G*T*T
70 LET X=X+3
```

O. — Astucieux !... Tu calcules le temps en fonction de X et tu reportes cette valeur dans la ligne 50 de calcul de Y !... D'ailleurs, cela ne change pratiquement rien à l'écran !

P. — Ceci dit, on va compliquer encore un peu en faisant intervenir la valeur de l'angle au départ !... En fait, cela ne rajoute qu'un nouveau terme dans le calcul de X et de Y, qui correspond à la projection sur l'axe considéré du vecteur de la vitesse de départ !

O. — Tu parlerais chinois que je ne comprendrais pas mieux !

P. — En clair, tu rajoutes au calcul de X le cosinus de l'angle au départ et au calcul de Y le sinus, comme cela :

$$x = V_0 t \cos A$$
$$y = V_0 t \sin A - \frac{1}{2} gt^2$$

O. — Si je comprends bien, il suffit que je modifie le programme de la page 127 comme suit :

```
40 LET X=V*T*COS (A)
50 LET Y=V*T*SIN (A)-G*T*T
```

P. — C'est ça, en n'oubliant pas la ligne qui détermine la valeur de l'angle A qui, je te le rappelle, doit être exprimée en radians !... Si, comme je le crois, tu préfères parler en degrés, il te faut alors écrire la ligne suivante de conversion :

```
5 LET A=45*PI/180
```

O. — Tiens, il y a un truc bizarre que je ne comprends pas !... J'ai mis le même angle que tout à l'heure, c'est-à-dire 45 degrés, et j'obtiens sur l'écran une courbe toute petite !... Normalement, je devrais avoir la même.

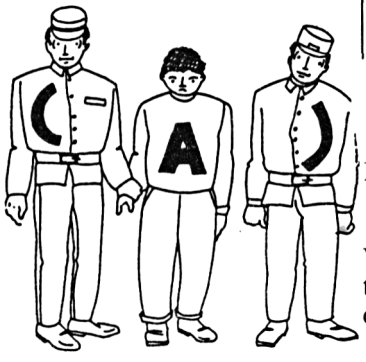
P. — Je vois d'où cela provient !... C'est dû au fait que le sinus et le cosinus de 45 degrés sont égaux tous les deux à 0,707, et que cela vient te diminuer les valeurs correspondantes de X et de Y !... Pour remédier à cela, porte donc la valeur de V à 1,414, car 1,414 × 0,707 donne un résultat égal à 1.

```
10 LET V=1.414
```

O. — Ok, c'est bon !... Tu me montres maintenant comment on fait pour faire varier X à la place de T !

P. — Tout de suite !... Voici les lignes qu'il faut modifier :

```
30 LET X=0
40 LET T=X/(V*COS (A))
50 LET Y=X*TAN (A)-G*T*T
70 LET X=X+3
```



O. — Tiens, tu as placé à la ligne 40 l'opération V*COS (A) entre parenthèses !... Pourquoi ?

P. — Parce que sans les parenthèses, l'ordinateur n'effectue que la division de X par V. Il n'effectue jamais une multiplication après une division, mais toujours avant !... A toi donc à écrire l'opération dans le bon ordre, ou alors à mettre des parenthèses là où il en faut !

O. — Bien !... Et à quoi correspond l'instruction TAN que tu as employée à la ligne 40 ?

P. — Il s'agit d'une instruction qui calcule la tangente de l'angle A et qui est égale au rapport du sinus sur le cosinus de cet angle !

O. — Ah bon !... Et pour le vent, comment on fait ?

P. — On le simule en introduisant dans le calcul de la coordonnée horizontale X un paramètre dans le genre : $x = V_0t + kt^2$.

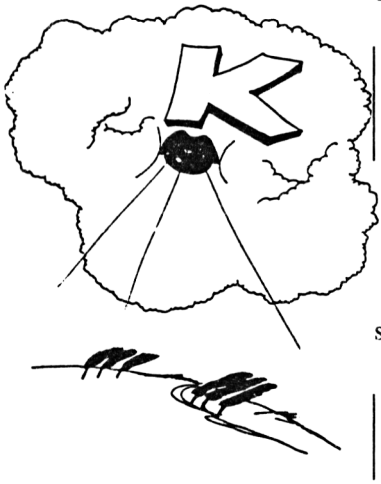
O. — Je suppose que c'est la valeur du paramètre K qui va déterminer la force du vent ?

P. — Exact !... Et c'est son signe qui va indiquer sa direction !

O. — Dans le fond, ça ne me paraît pas plus compliqué que la pesanteur !... Je vais essayer de modifier le programme moi-même !... Pour cela, il faut que je commence par écrire la ligne qui définit la valeur de K !... Au fait, je lui donne quelle valeur ?

P. — Essaye .01 ou .02 !... Tu verras bien ce qui convient le mieux !... Essaye aussi des valeurs négatives pour voir l'effet que donne un vent contraire !

O. — Très bonne idée !... Voici donc ce qui devrait correspondre à un vent arrière qui est censé pousser la balle :




```
 2 LET K=.02
40 LET X=V*T*COS (A)+K*T*T
```

P. — Et alors ?

O. — Ça pour pousser, il pousse !... Il pousse même tellement fort que la balle est sortie de l'écran !... Je vais essayer avec K=.01 !... Cela devrait mieux aller :

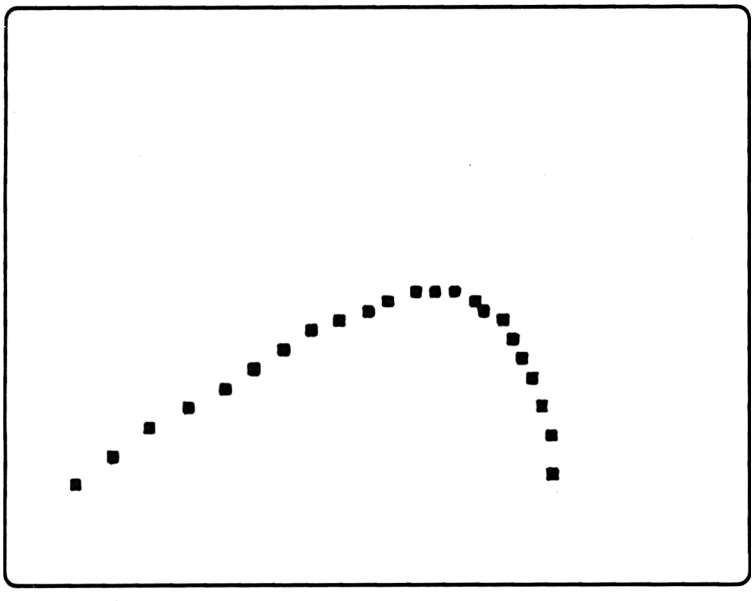
```
 2 LET K=.01
```

P. — Cela donne de bien meilleurs résultats, effectivement !... Essaye maintenant avec un vent contraire !

O. — Ok !... Je vais donc rajouter un signe moins à la valeur de K :

```
2 LET K=-.Ø1
```

P. — Ah, ce n'est pas mal !... On obtient une belle courbe ! Elle est même encore plus belle avec V = 2.



O. — Tu me laisses m'amuser avec un moment !... J'ai envie de changer l'angle et la vitesse de départ pour voir ce que cela donne !

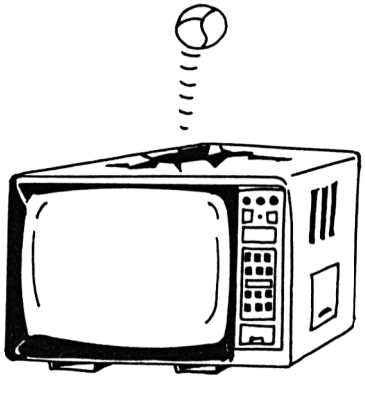
P. — Et comment donc !

Et je laissais Olivier devant son écran en train de se bagarrer avec les paramètres du tir balistique, bien décidé pour ma part à profiter de ce qui pouvait rester de ce samedi pas comme les autres !... Le répit fut de courte durée, interrompu qu'il fut par un rappel à l'ordre en provenance du haut :

O. — Dis-donc, Papa, j'ai quelques problèmes !

P. — Raconte-moi tout !

O. — Quand je tire presque à la verticale, le projectile sort de l'écran et cela interromp aussitôt le déroulement du programme !... Il n'aurait pas moyen de faire en sorte que le programme de calcul de la trajectoire continue à fonctionner, même après la sortie du projectile de l'écran, de façon à le revoir lorsqu'il rentre à nouveau dedans ?



P. — Si fait !... C'est même très simple !... Il te suffit d'écrire à la place de la ligne 6Ø :

```
6Ø IF Y<43 THEN PLOT X,Y
```

O. — Mais oui !... J'aurai dû y penser !... Et aussi, comment fais-tu pour effectuer la conversion de T en X ?

P. — Je ne fais pas, car pour obtenir la valeur de T en fonction de X, il faut résoudre une équation du second degré et cela, je n'en ai plus envie à cette heure-ci !

O. — Je me disais bien...

AUX COMMANDES DU CLAVIER

Tout au long de la semaine qui suivit, je sentais Olivier insatisfait de ce qu'il venait d'accomplir en informatique. Le samedi matin, je le questionnais à ce sujet :

P. — Alors, Olivier, qu'est-ce qui ne va pas ?

O. — A bien y réfléchir, tu sais, je ne trouve pas ça très drôle de dessiner des jeux vidéo à l'écran !

P. — Tiens donc !... Et c'est toi qui dis cela !

O. — Eh oui !... J'imaginais ça plus « marrant » !

P. — Dis plutôt que tu imaginais cela plus facile !... En fait, ce que tu ressens en ce moment, ce n'est rien d'autre que du découragement !... C'est un peu comme si tu escaladais le flanc d'une montagne et qu'arrivé à mi-chemin, tu regardes le trajet qu'il te reste à parcourir et que tu évalues les efforts que cela va te coûter. En procédant ainsi, tu prends conscience de l'aspect négatif de l'épreuve. Si au contraire tu te retournes et que tu regardes le trajet immense que tu as accompli depuis le bas de la vallée, tu prends alors conscience de l'aspect positif. En toutes choses, tu as cette dualité de polarités. A toi de saisir celle qui est constructrice, la positive, et de délaisser l'autre qui est destructrice !

O. — Oui, bien sûr !

P. — Pas convaincu, n'est-ce pas ?... Regarde donc un peu ce que tu sais faire !... Savais-tu, ne serait-ce qu'il y a une semaine, faire bouger un engin d'un bord à l'autre de l'écran ?

O. — Non !

P. — Et pourtant, cela te paraît tout simple maintenant !

O. — Elémentaire, je dirais même !

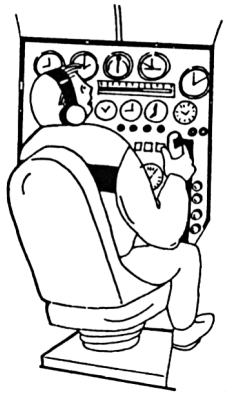
P. — Et que dirais-tu si je te rappelais qu'il n'y a guère plus d'un mois, j'ai dû t'apprendre à dessiner une ligne droite toute ordinaire !

O. — Tu as raison !... Je dirais que c'est ridicule !

P. — Tu vois bien !... Eh bien, tu en es dans ta progression vers les sommets des jeux vidéo, comme l'alpiniste dont je te parlais tout à l'heure en était de son ascension : au milieu !... Il te reste encore autant à faire que ce que tu viens d'accomplir. Et n'oublie pas que tu es accompagné d'un guide et qu'à deux, c'est toujours plus facile.

O. — Je sais bien !... Tu vois, ce qui me chagrine, c'est que je voudrais pouvoir concevoir tout seul les jeux que j'imagine et je me rends compte que je n'y arrive pas !

P. — Patience, patience !... Ça viendra !... Quand on a fait le parcours avec un guide, on est ensuite capable de le faire tout seul et, même, on n'hésite plus à emprunter des voies nouvelles !



O. — Bon !... Alors, en attendant que ça vienne, qu'est-ce qu'on étudie aujourd'hui ?

P. — La façon de donner les ordres qui vont te permettre de contrôler l'évolution de tes engins, un peu comme si tu étais assis dans le poste de pilotage et que tu tiennes en main les commandes !

O. — Whaooo !... Ça va être super ça !

P. — C'est bien ce que je me disais !... Je te propose donc de commencer par la commande de montée ou de descente d'un mobile.

Les commandes de montée et de descente

O. — C'est moi qui choisis l'engin !

P. — D'accord !... Qu'est-ce que tu prends ?

O. — Un avion !

P. — Bien !... On va donc faire voler l'avion de gauche à droite de l'écran, en le faisant réapparaître après chaque passage !

O. — Aucun problème, ça, je sais faire !... Il suffit d'écrire :

```
10 FOR C=0 TO 27
20 PRINT AT 10,C;" ▙▟-"
30 NEXT C
40 IF C>27 THEN PRINT AT 10,C;
"    "
50 GOTO 10
```

P. — Ok !... Cependant, comme l'avion va trop vite, diminue un peu sa vitesse avec l'instruction STEP !

O. — Voilà, c'est fait :

```
10 FOR C=0 TO 27 STEP .5
```

P. — C'est mieux ainsi !... Maintenant, on va poser une variable qui va déterminer son altitude !... Ainsi, en faisant varier la valeur de cette variable, on fera monter ou descendre l'avion !

O. — Compris !... Et une variable d'altitude qui marche, pardon, qui vole, une !... Son introduction m'oblige toutefois à modifier les lignes 20 et 40 comme suit :

```
 5 LET L=10
20 PRINT AT L,C;" ▙▟-"
40 IF C>27 THEN PRINT AT L,C;"
   "
```

P. — Bien !... Maintenant, on va commencer par faire descendre l'avion en appuyant sur la touche [6] !

O. — Pourquoi la 6 ?

P. — Tout simplement parce qu'il y a une flèche dirigée vers le bas dessinée dessus !... Bien entendu, tu peux choisir n'importe quelle autre touche, mais celle-ci est la plus évidente !

O. — Et comment détecte-t-on le fait qu'on appuie ou non sur cette touche ?

P. — A l'aide de l'instruction INKEY$!

O. — Quel nom barbare pour une instruction !

P. — Pas tant que cela !... En anglais « In key » veut dire « taper sur une touche du clavier » !... Il suffit alors de faire suivre cette instruction de l'indication de la touche, voire de plusieurs touches, que tu as sélectionnées pour déclencher une opération, en n'oubliant pas de placer les touches choisies entre guillemets !

O. — Quelque chose comme : IF INKEY$ = "6" ?

P. — Tout à fait !... Il ne te reste plus ensuite qu'à préciser la nature de l'opération que tu veux voir exécutée !... Dans le cas présent, comme tu veux faire descendre l'avion, il te faut écrire : IF INKEY$ = "6" THEN LET L = L + 1 !

O. — Et où place-t-on une telle instruction ?

P. — A l'intérieur de la boucle qui fait avancer l'avion, de façon que celui-ci puisse descendre d'une ligne chaque fois qu'il avance d'un carreau !

O. — Bon !... Alors voici la descente :

```
15 IF INKEY$="6" THEN LET L=L+
1
```

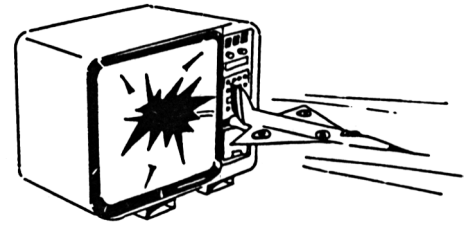
O. — Zut !... Chaque fois que je fais descendre l'avion, il reste une image de lui sur la ligne qu'il occupait avant !...

P. — Cela provient du fait que tu n'effaces pas l'emplacement précédent qu'il occupait !... Essaye donc avec la ligne suivante :

```
17 PRINT AT L-1,C;"      "
```

O. — Ok !... Maintenant, c'est bon !... Enfin presque !... Quand je reste appuyé trop longtemps sur la touche [6], l'avion descend trop bas et sort de l'écran, ce qui bloque le déroulement du programme !

P. — C'est normal, nous n'avons pas mis de butée inférieure !... Il faut donc rajouter dans la ligne 15 :



```
15 IF INKEY$="6" AND L<21 THEN
LET L=L+1
```

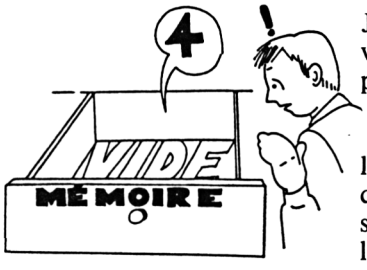
O. — Je comprends !... Ainsi, pour que l'avion descende, il faut que deux conditions soient remplies : que la touche [6] soit enfoncée et qu'il ne soit pas déjà sur la ligne 21 !

P. — Bien !... Alors, puisque tu as compris, je vais te laisser écrire les lignes d'instructions qui vont te permettre de le faire monter !

O. — Facile !... Les voici :

```
16 IF INKEY$="7" AND L>Ø THEN
LET L=L-1
18 PRINT AT L+1,C;"      "
```

P. — Bravo !... Tu essayes pour voir ce que cela donne !



O. — Ça marche !... C'est super !... Tiens, c'est moins super que je pensais !... J'obtiens une saturation de la mémoire quand l'avion vole dans le bas de l'écran !... Je ne vois pas pourquoi elle sature, cette fichue mémoire !... Ce n'est quand même pas un si petit avion qui en est la cause ?

P. — Eh si !... Ce phénomène de saturation provient de l'effacement du sillage de l'avion. Tu n'y as peut-être pas prêté attention, mais au fur et à mesure qu'il se déplace dans l'écran, tu remplis la mémoire d'affichage d'espaces blancs. Or, les espaces blancs sont des caractères comme les autres et l'ordinateur n'est pas capable d'aller vérifier dans la mémoire si ce que tu y as inscrit est l'un des 255 caractères ou bien un espace blanc. Quand il constate que la mémoire d'affichage est pleine, il envoie le message d'erreur 4/... que tu connais bien, même si l'écran est plein de vide !

O. — Quel "couillon" !

P. — Ce n'est qu'une machine, tu sais !

O. — Alors, comment faire ?

P. — Changer de méthode d'effacement !

O. — J'y suis !... Il faut que, lorsque l'avion arrive en bout de l'écran, j'efface globalement et l'avion et le sillage invisible qu'il laisse derrière lui, en employant l'instruction CLS, à la place de la ligne 4Ø :

```
4Ø IF C>27 THEN CLS
```

P. — Et qu'est-ce que cela donne ?

O. — C'est bon !

P. — Alors, puisque c'est bon, je vais profiter de l'occasion pour te rappeler ce que je t'avais dit concernant la position de l'instruction CLS !

O. — Vas-y, je t'écoute !

P. — Prends l'habitude de placer une telle instruction en début de programme et non à la fin comme c'est le cas présentement !... Tu y gagneras sur tous les plans, que ce soit en économie de caractères ou en diminution de l'effet de clignotement qui en résulte !

O. — Tu veux dire que si je déplace l'instruction CLS pour la mettre en tête de programme, j'économise des caractères ?

P. — Bien sûr !... En plaçant l'instruction CLS en tête de programme, tu peux carrément supprimer ta ligne 4Ø, à condition de modifier ta ligne 5Ø comme ceci :

```
 9 CLS
50 GOTO 9
```

O. — Effectivement, il faut moins de caractères pour obtenir le même résultat !...

P. — Quant à la position de l'instruction CLS qui efface tout, par rapport à la ligne 20 qui imprime l'avion, tu gagnes une ligne d'instruction !

O. — Compris !... Dis-donc, c'est drôlement bien maintenant !... Je peux faire monter et descendre l'avion sans qu'apparaisse la moindre saturation !... Vive l'instruction CLS !... Tiens, ça ne marche plus !

P. — Qu'as-tu comme message d'erreur ?

O. — 5/18 !

P. — Alors, je comprends ce qui c'est passé !... A la ligne 18, tu demandes à l'ordinateur d'effacer en-dessous de l'avion, c'est-à-dire à la ligne L + 1. Or, quand tu es descendu jusqu'à la ligne 21, tu lui demandes d'effacer sur la ligne 22, et ça, il ne peut le faire !

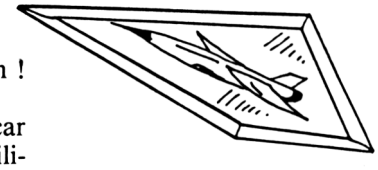
O. — Décidément, qu'est-ce que c'est compliqué tout ça !... Et qu'est-ce que je dois faire pour supprimer ce défaut ?

P. — Il suffit de remonter d'une ligne ta butée inférieure, en modifiant la ligne 15 ainsi :

```
 15 IF INKEY$="6" AND L<20 THEN
LET L=L+1
```

O. — Ça marche, mais il y a quelque chose que je ne comprends pas !... Pourquoi n'y a-t-il pas le même défaut en haut de l'écran ?

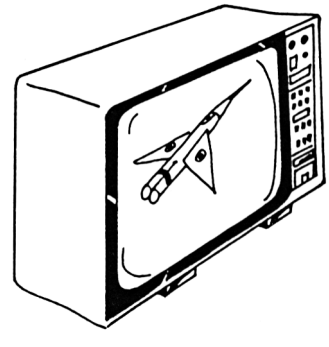
P. — Il y est, mais tu ne le vois pas !... Le haut de l'écran se comporte différemment du bas !... En fait, il suffit que tu le considères comme un miroir !... Ainsi, lorsque l'avion est sur la ligne 0, l'ordre d'effacement qui devrait se produire sur la ligne − 1, se produit sur la ligne + 1, ce qui ne bloque pas le déroulement du programme !



O. — Si je me souviens bien, cela se produisait déjà avec le bord gauche de l'écran !

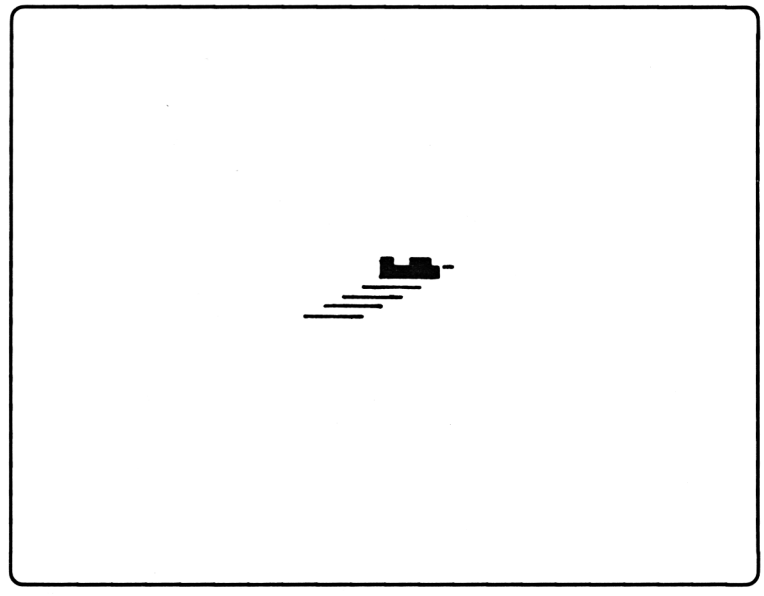
P. — Exact !... C'est le même phénomène !... Tu peux donc, si tu le souhaites, car ce n'est pas indispensable, modifier ta ligne 16 pour qu'elle présente une certaine similitude avec la 15 !

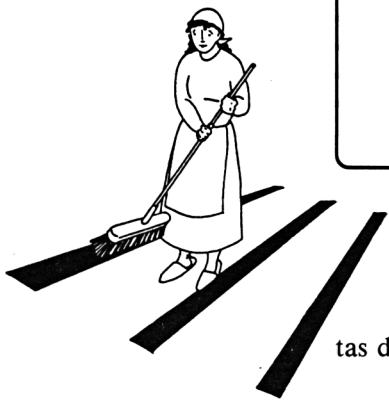
O. — Voilà, c'est fait !



```
 16 IF INKEY$="7" AND L>1 THEN
LET L=L-1
```

P. — Tu vois qu'on y est arrivé !... J'espère que le résultat final te convient ?





O. — Oui, quoique je trouve le déplacement de l'avion trop lent !

P. — C'est le résultat de l'instruction STEP .5 !... Maintenant qu'on a rajouté des tas de lignes d'instruction dans la boucle, on peut la supprimer !

O. — Ok !... Je vais en profiter pour renuméroter chaque ligne, de façon à y voir plus clair :

```
   5 LET L=10
  10 CLS
  20 FOR C=0 TO 27
  30 IF INKEY$="6" AND L<20 THEN
 LET L=L+1
  40 IF INKEY$="7 AND L>1 THEN
LET L=L-1
  50 PRINT AT L-1,C;"     "
  60 PRINT AT L+1,C;"     "
  70 PRINT AT L,C;" ▟▙-"
  80 NEXT C
  90 GOTO 10
```

P. — Effectivement, cela ne fait pas de mal, un peu d'ordre !

O. — Et maintenant, qu'est-ce qu'on fait ?

La commande de déplacement latéral

P. — Il me semble que l'étude de la commande de déplacement latéral s'impose !... Qu'en penses-tu ?

O. — Bonne idée !

P. — Cette fois, par contre, c'est moi qui choisis l'engin et c'est toi qui fais le programme !

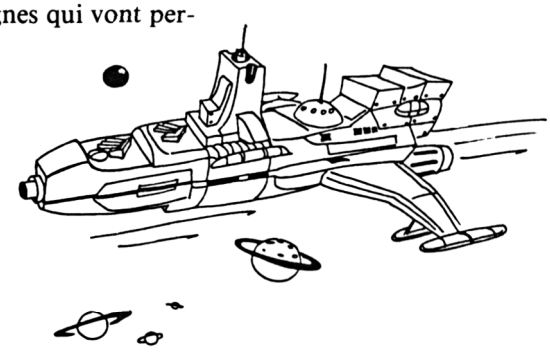
O. — Bon, d'accord !... Qu'est-ce que tu choisis ?

P. — Un véhicule spatial que je vais manœuvrer dans un champ d'astéroïdes !

O. — Super !... C'est donc à moi de débuter !... Je vais de ce pas placer l'engin au centre de l'écran en fixant ses coordonnées de départ :

```
10 LET L=10
20 LET C=16
```

P. — Bien !... Je suppose maintenant que tu vas écrire les deux lignes qui vont permettre le déplacement latéral de l'engin vers la gauche ou la droite ?



O. — Oui, bien sûr !... Voici déjà la première :

```
30 IF INKEY$="5" AND C>0 THEN
LET C=C-1
```

P. — Mes compliments !... Elle est bien écrite du premier coup !

O. — Qu'est-ce que tu crois ?... J'apprends mes leçons !... Par contre, pour écrire la seconde ligne, j'aimerais connaître les dimensions de ton véhicule spatial !

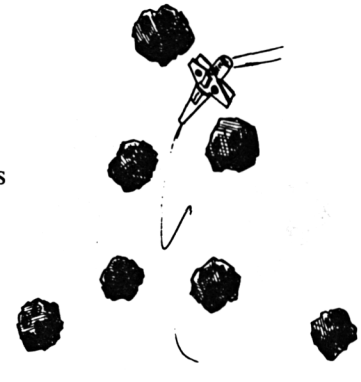
P. — Considère qu'il fait deux carreaux de longueur sur un seul carreau de hauteur !

O. — Merci !... Comme il faut que j'efface de chaque côté du véhicule, il faut donc que je tienne compte de quatre carreaux au total pour le calcul de la butée de droite !... Voici donc la seconde ligne :

```
40 IF INKEY$="8" AND C<28 THEN
LET C=C+1
```

P. — Ok !... Et comme prévu, voici l'engin spatial :

```
50 PRINT AT L,C;" ▚▞ "
```



O. — Il est « marrant », il est tout petit !

P. — C'est pour mieux se frayer un chemin parmi les astéroïdes !

O. — C'est vrai que c'est pas facile, tellement ils sont rapprochés les uns des autres !... En attendant, je termine mon petit programme par l'instruction de boucle :

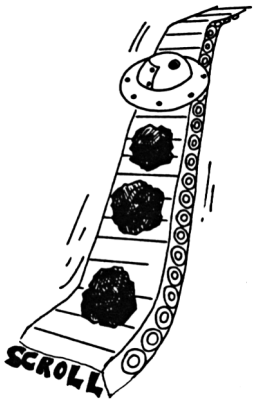
```
60 GOTO 30
```

P. — Et voilà une commande de déplacement latéral qui fonctionne parfaitement !

O. — Bien, maintenant je passe au programme d'astéroïdes !... Il n'est pas dur à écrire, mais je ne sais pas où l'insérer dans le programme actuel !

P. — Alors, si tu veux bien, nous allons réfléchir ensemble sur ce problème !... Je suppose qu'après avoir écrit la ligne qui dessine au hasard les astéroïdes, tu vas employer l'instruction SCROLL ?

P. — Oui, bien sûr !



O. — Or, tu n'ignores pas que cette instruction va te faire monter à la fois les astéroïdes et l'engin spatial !

P. — C'est vrai, ça !

O. — Il va donc falloir que tu effaces l'engin qui aura grimpé d'une ligne, aussitôt après la ligne d'instruction SCROLL !

O. — J'ai compris !... Comme l'écran sera à ce moment-là vide de tout engin, il faut que cette ligne d'effacement précède la ligne de réimpression du véhicule de façon à éviter tout clignotement !... J'écris ça tout de suite :

```
42 PRINT AT 21,INT (RND*32);"*
"
45 SCROLL
47 PRINT AT L-1,C;"      "
```

P. — Tu vois qu'en fin de compte il n'y avait pas d'autre choix possible que celui-là !

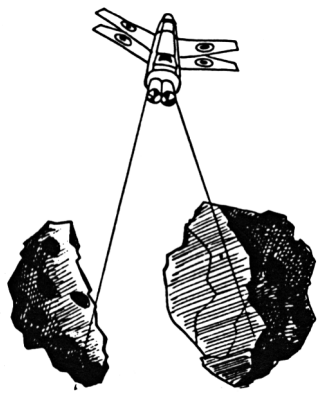
O. — Oui, oui !... Dis-donc, tu sais qu'il est super ce jeu !... Il faut que je slalome entre les astéroïdes qui m'arrivent dessus, de façon à les éviter !

P. — Tu vois, tu te crois déjà aux commandes de ton vaisseau spatial !

O. — Eh oui !... Tiens, je viens de découvrir un petit défaut à ce programme !

P. — Ah bon !... Et lequel ?

O. — J'efface tous les astéroïdes que je frôle !



P. — Ok, je vois d'où cela provient !... C'est l'espace blanc d'effacement que j'ai placé à droite et à gauche du vaisseau qui nous joue ce tour !...

O. — Ça, pour effacer, il efface tout !... Comment peut-on supprimer ce défaut ?

P. — Je ne vois qu'une solution : supprimer les deux espaces d'effacement !

O. — Bien sûr mais, dans ce cas, c'est l'effacement du déplacement latéral qui ne se fait plus !

P. — C'est pourquoi il faut employer une autre méthode !... Celle que je te propose consiste à effacer carrément tout le vaisseau, une ligne avant les lignes de commande du déplacement latéral !... Comme tu le réimprimes aussitôt après ces deux lignes, il ne va pas rester invisible trop longtemps !... Cela va te le faire clignoter légèrement, mais tu verras que ce phénomène ne fait que rajouter du réalisme au jeu !

O. — L'idée me semble bonne !... Laisse-moi modifier tout seul mon programme !... Voilà déjà la ligne d'effacement que je place avant les lignes 30 et 40 de commande :

```
25 PRINT AT L,C;"   "
```

O. — ... et voici les lignes 47 et 5Ø modifiées en conséquence :

```
47 PRINT AT L-1,C;"   "
5Ø PRINT AT L,C;"▟▙"
```

P. — Sans oublier la ligne 4Ø dont la valeur de la butée de droite peut être augmentée de 2 carreaux :

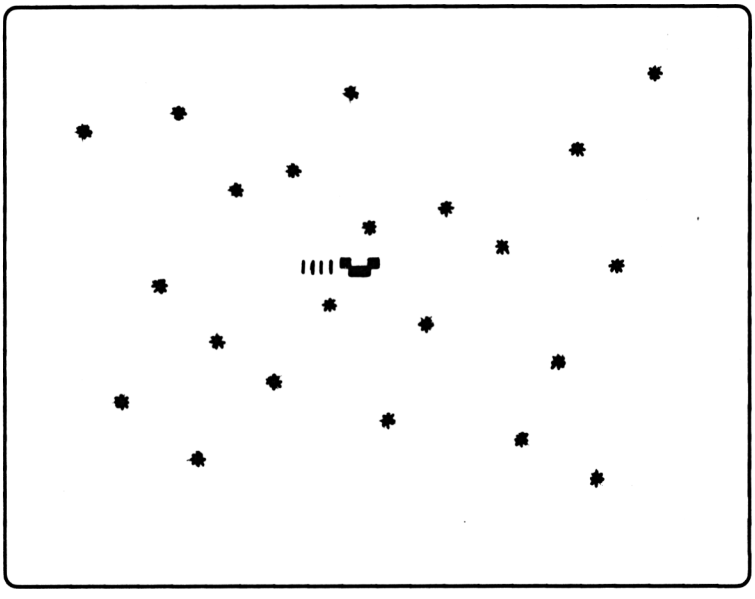
```
4Ø IF INKEY$="8" AND C<3Ø THEN
LET C=C+1
```

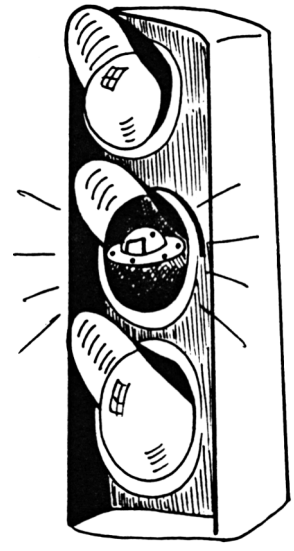
O. — ... et la ligne 6Ø qu'il faut modifier si tu veux que ça marche :

```
6Ø GOTO 25
```

P. — Je l'avais oubliée, celle-là !... Tiens, je viens de découvrir quelque chose d'intéressant en relisant le programme que nous venons d'écrire !

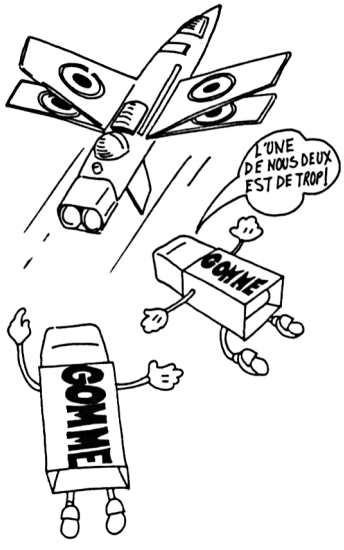
O. — Qu'est-ce que c'est ?





P. — On efface deux fois le vaisseau spatial, une fois à la ligne 47 parce que l'instruction SCROLL l'a fait remonter d'une ligne et une autre fois à la ligne 25 avant les commandes de déplacement !... Il ne te semble pas que c'est une fois de trop ?

O. — Peut-être...



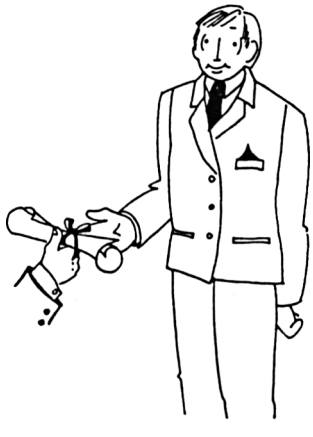
P. — Suis bien mon raisonnement !... On efface le vaisseau à la ligne 47 parce qu'il a monté d'une ligne : c'est normal !... Ensuite on le redessine à la ligne 5Ø pour aussitôt le réeffacer à la ligne 25 !... C'est pas logique tout ça !

O. — Ça, c'est vrai !... Et pourtant, la ligne 25 est indispensable à cause du déplacement latéral !

P. — Pas si tu places ces deux lignes de commande entre les lignes 47 et 5Ø. Dans le fond, peu t'importe d'effacer le vaisseau une ligne plus haut sur l'écran, pourvu que tu l'effaces !

O. — Très astucieuse ton idée !... Bouge pas, j'essaye tout de suite pour voir si elle est bonne :

```
  1Ø LET L=1Ø
  2Ø LET C=16
  3Ø PRINT AT 21,INT (RND*32);"*
"
  4Ø SCROLL
  5Ø PRINT AT L-1,C;"  "
  6Ø IF INKEY$="5" AND C>Ø THEN
LET C=C-1
  7Ø IF INKEY$="8" AND C<3Ø THEN
 LET C=C+1
  8Ø PRINT AT L,C;"▚▞"
  9Ø GOTO 3Ø
```



P. — Et alors ?

O. — Elle l'est !

P. — Te rends-tu compte que tu as presque réalisé tout seul ton premier jeu vidéo ?

O. — C'est vrai !... Il n'y manque que la détection des collisions entre le vaisseau spatial et les astéroïdes pour qu'il soit complet !

P. — Cela, nous nous en occuperons la semaine prochaine !... Pour le moment, je te suggère de continuer à examiner avec moi les différentes commandes auxquelles tu es susceptible de devoir faire appel au cours de l'élaboration d'un jeu !

O. — Ok !... Quelle est donc la suivante ?

P. — Il me semble que la commande de changement de vitesse ne serait pas mal !

La commande de changement de vitesse

O. — Bonne idée !... Et par quoi on l'illustre ?

P. — Je ne sais pas trop !... Que dirais-tu de faire atterrir un hélicoptère sur le toit d'un immeuble ?

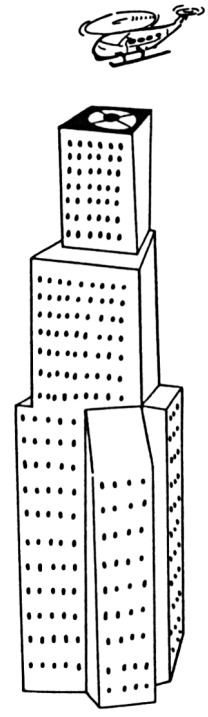
O. — Mouais !... Ça ne me paraît pas aussi super chouette que le jeu précédent, mais on fera avec !

P. — En tout cas, ce que je puis t'assurer, c'est que ce ne sera certainement pas facile à programmer !... Il va falloir combiner à la fois les commandes de montée et de descente, ainsi que les commandes de marche avant et de marche arrière avec celles de changement de vitesse !

O. — Sans parler de la mise en route du moteur et de la rotation des pales !

P. — Bien sûr... En attendant d'en arriver là, voici déjà l'immeuble dont le toit va nous servir de piste d'atterrisage :

```
10 CLS
20 PRINT AT 21,16;"▟▟▟▟ "
```



O. — Pourquoi tu ne le dessines pas plus haut, ton immeuble ?

P. — Pour des raisons que tu connais bien, mais que je vais te rappeler quand même !... Comme l'hélicoptère que tu vas dessiner va se promener dans le ciel de ton écran, il va laisser un sillage derrière lui...

O. — ... qui va saturer la mémoire si on ne prend pas la précaution d'effacer de temps en temps l'écran avec l'instruction CLS !

P. — C'est pourquoi je l'ai placée en début de programme !... Or, tu sais qu'après une telle instruction, il faut réécrire immédiatement les structures fixes, de façon à minimiser l'inévitable effet de clignotement qui en découle !

O. — Oui, je me souviens !... C'est donc pour ça que tu limites la hauteur de l'immeuble à une ligne seulement ?

P. — Exactement !... Toutefois, pour te montrer ce que cela donne lorsqu'il y a plusieurs étages à un immeuble, je te propose le petit programme suivant qui va simuler un programme de jeu, et grâce auquel tu vas pouvoir vérifier l'effet de clignotement que procure une structure fixe de hauteur variable :

```
10 CLS
20 LET H=10
30 FOR L=22-H TO 21
40 PRINT AT L,16;"▟▟▟▟ "
50 NEXT L
60 FOR A=0 TO 50
70 NEXT A
80 GOTO 10
```

O. — Effectivement, ce n'est pas brillant !... Et même en réduisant la hauteur, ce n'est guère mieux !... Tu avais raison de la limiter à une seule ligne !... Au-delà, l'effet de clignotement devient trop visible !

P. — Bien !... Ceci étant établi, je te propose maintenant de dessiner l'hélicoptère !

O. — Attends, je viens d'avoir une idée !... Pourquoi ne pas le faire décoller du pont d'un navire, plutôt que du toit d'un immeuble !... Ce sera beaucoup plus drôle...

P. — ... surtout lorsque le navire se déplace en pleine mer et qu'il y a du vent !... Très bonne idée !

O. — Je te remplace donc ton affreux immeuble par le superbe navire porte-hélicoptères que voici, sans oublier la mer sur laquelle il vogue :

```
10 CLS
20 PRINT AT 21,0;"----------
--------------------"
30 PRINT AT 21,10;"▬▬▬
```

d'effacement juste avant la ligne d'impression de l'hélicoptère, ce qui supprime tout clignotement !

O. — Comme cela ?

```
85 PRINT AT L-2,C;"    ";AT L+1
,C;"    "
```

P. — Vois par toi-même !

O. — Ça ne me paraît pas génial !... On n'a pas l'impression que l'hélicoptère vole comme tout à l'heure !... En plus, quand je le fais apponter sur le navire, il efface une partie de celui-ci !

P. — La solution à ce petit problème n'est guère compliquée !... Il suffit que tu sépares ta ligne d'instruction 85 en deux lignes distinctes, et que tu interdises l'effacement en dessous de l'hélicoptère de fonctionner lorsque L = 20, c'est-à-dire lorsqu'il est posé sur le pont ou qu'il vole au ras de l'eau.

O. — J'ai compris !... Quelque chose comme :

```
85 PRINT AT L-2,C;"    "
87 IF L<20 THEN PRINT AT L+1,C
;"    "
```

P. — C'est ça !... Tu vois, maintenant, l'effacement ne se fait plus lorsque l'hélicoptère est sur la ligne 20 !

O. — Oui, c'est bien !... N'empêche que je reprends ma première solution, car elle est meilleure !

P. — Comme tu voudras !... On continue par le déplacement du bateau ?

O. — Bien sûr !... Pour cela, il faut commencer par poser une variable à la place de la coordonnée de colonne. Ensuite, il faut définir la valeur initiale de cette variable, puis rajouter un signe moins et un espace à l'arrière du bateau, écrire ensuite la ligne d'augmentation de valeur de la variable X, modifier la ligne 100 et prévoir la sortie d'écran !... Tu vois que je l'ai bien apprise, ma leçon !

```
 9 LET X=0
30 PRINT AT 21,X;"-[redacted]";AT
20,X+1;" ▟"
95 LET X=X+1
97 IF X=26 THEN GOTO 9
100 GOTO 30
```

P. — Il faudrait être difficile !... Mais, trève de compliments, il est temps de passer au plat de résistance, à savoir la commande de vitesse de l'hélicoptère !

O. — Je suis prêt !

P. — Cette commande va être différente de celle du déplacement latéral que nous avons vue précédemment, en ce sens que les deux touches 5 et 8 vont agir respectivement

comme frein ou accélérateur, tout en combinant également la marche arrière et la marche avant !

O. — Je crois comprendre ce que tu veux dire !... Si on suppose que l'hélicoptère vole en avant à la même vitesse que le navire, un coup sur la touche [8] va accélérer sa vitesse, alors qu'un coup sur la touche [5] va le mettre en vol stationnaire, ou encore en marche arrière si on appuie une seconde fois sur cette même touche !

P. — C'est tout à fait cela !... En fait, tu vas faire avancer l'hélicoptère d'une quantité A qui peut être égale à −1, Ø, +1 ou +2, selon que ce dernier est en marche arrière, en vol stationnaire, en marche avant lente ou en marche avant rapide.

O. — Génial !... Ainsi, si je veux le faire apponter sur le navire, il faut que leurs vitesses soient identiques, de façon qu'il apparaisse immobile par rapport au navire !

P. — Sais-tu que tu viens de redécouvrir la relativité...

O. — Peut-être !... En attendant, c'est pas ça qui va me donner mon programme de changement de vitesse !... Pour commencer, il faut que je définisse la vitesse initiale de l'hélicoptère, et qu'ensuite je le fasse se déplacer de cette quantité à chaque parcours de la boucle !... Ça devrait aller avec les deux lignes suivantes :

```
 5 LET A=1
85 LET C=C+A
```

P. — Très bien !... Je note que tu as décidé de placer la ligne de progression de l'hélicoptère juste après les lignes de commande au clavier !...

O. — Oui, j'avais le choix de la placer juste avant ou juste après les lignes 5Ø à 8Ø !... J'ai choisi la seconde solution !... Par contre, je ne sais pas du tout comment faire pour détecter la sortie de l'hélicoptère de l'écran et commander sa réapparition sur le bord opposé, sans perturber le bateau !

P. — Petit problème !... Voici sa solution :

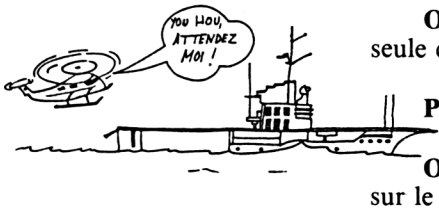
```
87 IF C=3Ø THEN LET C=Ø
```

O. — Et ça suffit ?

P. — Essaye donc !

O. — Effectivement, ça marche !... Mais comment donc s'effectue l'effacement ?

P. — Tout simplement grâce à la ligne 4Ø qui efface tout à la fois les déplacements horizontaux et verticaux, ainsi que les sorties d'écran en prime, à chaque parcours de la boucle !



O. — Que cette méthode d'effacement est donc puissante !... Désormais, c'est la seule que j'emploierai dant tous mes jeux !

P. — Tu ne remarques rien de particulier ?

O. — Si, j'allais justement t'en parler !... On dirait que l'hélicoptère prend du retard sur le bateau à chaque réapparition qu'il fait !... De quoi ça peut provenir ?

P. — De la différence de longueur entre les deux !... Si tu compares les lignes 97 et 87, tu vas constater que tu donnes l'ordre d'effacement du bateau à la colonne 26, et celui de l'hélicoptère à la colonne 3Ø. Il doit donc parcourir 4 colonnes de plus que le navire à chaque passage. C'est de là que provient son retard !

O. — Si ce n'est que ça !... Voilà ce qui devrait solutionner le problème :

```
87 IF C=3Ø THEN LET C=4
```

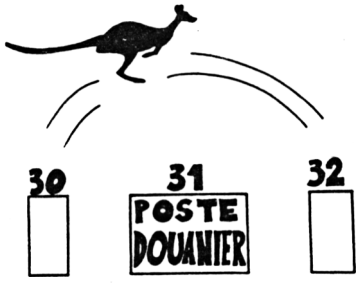
P. — Ce qui est effectivement le cas !

O. — Puisque tout marche bien, je termine par les deux lignes de commande du changement de vitesse :

```
5Ø IF INKEY$="5" AND A>-1 THEN
LET A=A-1
8Ø IF INKEY$="8" AND A<2 THEN
LET A=A+1
```

P. — Bien !... Tu essayes pour voir comment fonctionne ce programme, maintenant qu'il est terminé !

O. — Tout de suite, tu penses !... C'est chouette, je peut maintenant commander la vitesse de l'hélicoptère pour rattraper le navire !... Tiens, je viens d'obtenir un message d'erreur B/9Ø !... C'est dû à quoi, à ton avis ?



P. — Attends que je réfléchisse !... Ah, je viens de trouver !... Tu demandes à la ligne 9Ø d'imprimer l'hélicoptère dans la colonne 32 lorsque tu progresses de deux en deux !

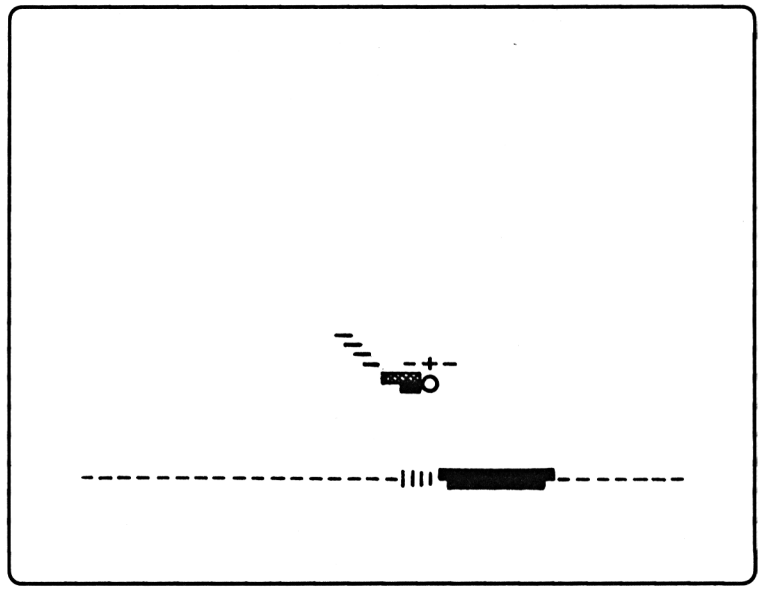
O. — Pourtant la ligne 87 devrait empêcher cela !

P. — Pas si tu progresses de deux carreaux à chaque fois !... Tu peux très bien être sur la colonne 29 et te retrouver le coup d'après sur la 31 !... La ligne 87 qui détecte la valeur 3Ø n'aura rien vu !

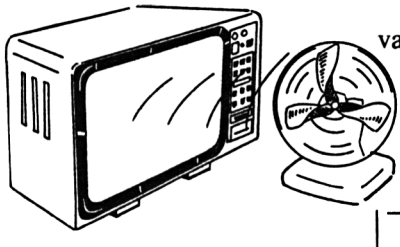
O. — Exact !... La solution est donc toute simple !... Il suffit de modifier cette ligne en remplaçant le signe = par >=, et le tour est joué :

```
87 IF C>=3Ø THEN LET C=4
```

P. — C'est bien !... Il ne te reste plus qu'à le faire atterrir sur le pont du navire !



O. — C'est déjà fait !... C'est beaucoup trop facile !



P. — Ah, tu trouves cela trop facile !... Attends un peu que je te rajoute du vent, tu vas voir !

O. — Oh oui, oh oui !

P. — Alors, puisque tu en veux, en voilà :

```
45 LET L=L+INT (RND*3-1)
```

O. — Il n'est pas génial ton vent, tu sais !... Il souffle aussi fort au ras de l'eau qu'en altitude !... Ce qu'il faudrait, c'est un vent qui soit nul au ras de l'eau et de plus en plus fort au fur et à mesure que l'on monte !

P. — S'il n'y a que cela pour contenter Monsieur !... Voici donc un vent qui croît proportionnellement à l'altitude :

```
45 LET L=L+(20-L)*INT (RND*3-1
)/10
```

O. — Pas mal, quoique je le trouve un peu trop mou à mon goût !

P. — Décidément, tu es contrariant aujourd'hui !... Si tu veux un vent plus fort, tu n'as qu'à remplacer le 10 par un 5 comme cela :

```
45 LET L=L+(20-L)*INT (RND*3-1
)/5
```

O. — Là, c'est bon !... Enfin, cela le serait si la mémoire ne saturait pas si vite !... Ah, celle-là !... Quand est-ce que tu m'achètes la mémoire 16K ?

P. — Je crois que cela ne va plus tarder !... En attendant, il faut trouver une solution !... Je vais t'en proposer une qui vaut ce qu'elle vaut, mais qui en tout cas supprime définitivement toute saturation !

O. — Dis vite !

P. — Il suffit d'effectuer un effacement complet de l'écran à chaque parcours de la boucle !... Tel que nous avons écrit notre programme, cela s'obtient facilement en modifiant la ligne 1ØØ ainsi :

```
100 GOTO 10
```

O. — Ah, je comprends !... Au lieu de renvoyer sur la ligne 3Ø à chaque fin de boucle, tu effectues un branchement direct à la ligne 1Ø qui comporte l'instruction CLS !... Mais, du coup, la ligne 4Ø d'effacement de l'hélicoptère ne sert plus à rien !

P. — Exact !... Il ne te reste plus qu'à la supprimer !

O. — Voilà, c'est fait !... Dans le fond, ça ne change pas grand chose !

P. — Si, quand même !... Ne remarques-tu pas que cela ralentit le jeu et que le bateau avance par saccades ?

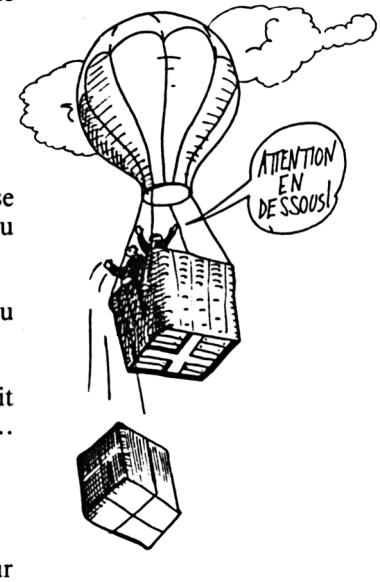
O. — Si, bien sûr !... Mais puisqu'on n'a pas d'autre solution !... Oh, je viens de penser à quelque chose !... Si on faisait tourner les pales de l'hélicoptère !

P. — D'accord, mais on terminera là-dessus !... Si tu te souviens, il suffit d'écrire une ligne qui ne représente que le rotor et que l'on place au milieu du programme !

O. — Vu qu'on imprime les pales à la ligne 9Ø et qu'on les efface à la ligne 1Ø, cela ne laisse pas beaucoup de choix !... Je pense que la ligne 98 serait la meilleure pour cela :

```
98 PRINT AT L-1,C;" + "
```

P. — C'est bien !... Mais je pense qu'une telle ligne est inutile avec la mémoire 1K, car le vol de l'hélicoptère est déjà suffisamment saccadé sans qu'il soit nécessaire d'en rajouter !... Par contre, elle donne un effet de rotation remarquable si tu possèdes une mémoire 16K et si tu n'effaces pas systématiquement avec CLS à chaque boucle.



La commande de largage

P. — Puisque nous en sommes au thème de la mer et des avions, je te propose d'illuster l'étude de la commande de largage par un avion qui lache un parachutiste au dessus d'une île.

O. — Et en quoi une telle commande diffère-t-elle de ce que nous avons déjà vu ensemble ?

P. — Elle en diffère par le fait qu'une fois que tu as appuyé sur la touche qui fait sauter le parachutiste de l'avion, le processus de la descente t'échappe complètement !... Tu initialises la descente et ensuite elle continue toute seule !

O. — Ah, c'est ça !... Alors, ça ne me paraît pas évident à réaliser !

P. — C'est justement pour cela que nous allons l'étudier tous les deux !... Pour commencer, tu vas dessiner une île et un avion qui circule dans le ciel !

O. — Ok !... Voici déjà pour l'île :

```
 10 CLS
 20 PRINT AT 21,0;"--------████
-------------------"
```

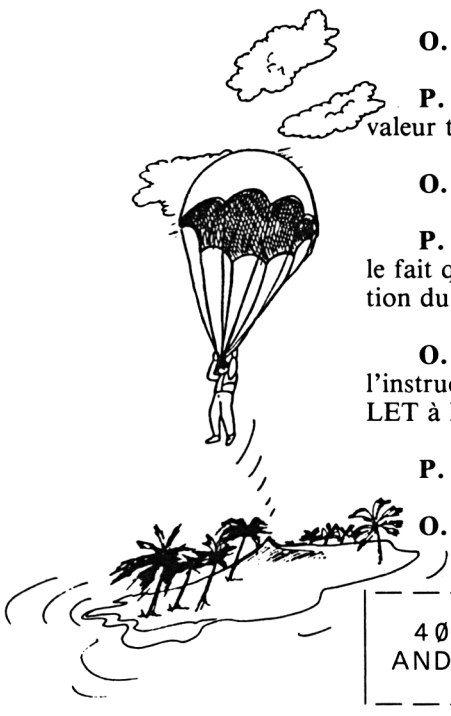
O. — ... et voici pour l'avion que je vais faire voler tout en haut de l'écran :

```
  9 LET X=0
 30 PRINT AT 1,X;" ██▄▄█";AT 0,X
;" ■ "
 80 LET X=X+1
 90 IF X=28 THEN GOTO 9
100 GOTO 30
```

P. — Je vois que tu as bien retenu la méthode que nous avons mise au point dans l'exercice précédent !

O. — Il n'y avait pas de raison d'en changer !... Quand une méthode marche...

P. — Bon !... Maintenant, c'est à moi de jouer !... Lorsque tu veux larguer un objet ou un personnage d'un engin qui se déplace, il faut réaliser deux opérations. En premier lieu, il faut initialiser la variable de la colonne de descente du parachute avec la valeur que possédait la variable de progression horizontale de l'engin. Ainsi, le parachute descendra dans la colonne sur laquelle passait l'avion au moment où tu as appuyé sur la touche de largage.



O. — Ça, c'est clair !... Et la seconde opération, c'est quoi ?

P. — Ensuite, il faut donner à la variable de descente verticale du parachute une valeur telle que ce dernier soit engagé dans un processus automatique de descente.

O. — Là, j'avoue que je ne vois pas !

P. — C'est le contraire qui me paraîtrait anormal !... La difficulté réside donc dans le fait qu'il faut commander deux opérations différentes à la suite d'une ligne d'instruction du genre : IF INKEY$ = "0" THEN LET...

O. — Et alors ?... Je ne vois pas où est le problème !... Il suffit d'écrire deux fois l'instruction LET à la suite de l'instruction INKEY$, en rattachant la seconde instruction LET à la première par une instruction AND !

P. — Ecris-la donc pour voir ce que cela donne !

O. — Tout de suite !... Voilà ce que je voulais dire :

```
 40 IF INKEY$="0" THEN LET C=X
AND ......
```

P. — Et la suite ?

O. — Je ne comprends pas !... Je n'arrive pas à écrire la seconde instruction LET !

P. — C'est normal, tu sais !... Il n'est pas possible d'écrire une seule des instructions placées au-dessus des touches après une instructions AND !... Donc, pas question d'écrire l'instruction LET !

O. — Mais alors, comment faire ?

P. — Tu viens de mettre en évidence l'une des faiblesses du langage BASIC employé. Il n'est pas possible de commander directement deux opérations après une instruction IF !... Il faut soit écrire deux lignes séparées en répétant à chaque fois :

```
40 IF INKEY$="Ø" THEN LET C=X
42 IF INKEY$="Ø" THEN LET ....
```

P. — ... soit faire appel à un sous-programme du genre :

```
40 IF INKEY$="Ø" THEN GOSUB 11
Ø
110 LET C=X
120 LET ......
130 RETURN
```

O. — Je préfère la seconde méthode du sous-programme !

P. — Moi aussi, je dois t'avouer, quoique la première méthode est la moins gourmande en mémoire !... Au passage, je vois que tu as déjà initialisé la variable de la colonne de descente en écrivant LET C = X !

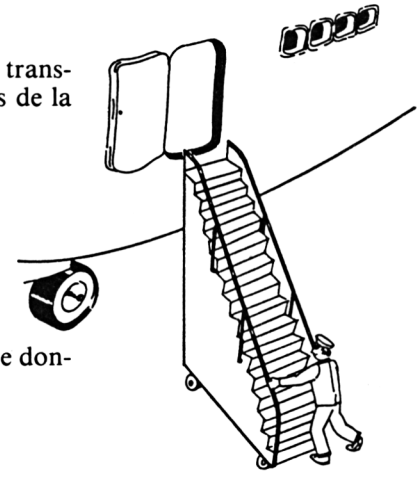
O. — Oui, puisque j'ai choisi C pour cela !

P. — Ensuite, il faut engager l'objet ou le personnage largué sur une sorte de tapis roulant vertical !... Cela s'obtient en déterminant une valeur de repos pour la coordonnée L de descente et en interdisant chaque ligne d'instruction d'être effective pour cette valeur de repos !

O. — Je commence à entrevoir une faible lueur de compréhension !... Tu veux bien préciser ta pensée ?

P. — On va prendre l'exemple du parachutiste pour cela !... L'avion qui le transporte vole sur la ligne n° 1. C'est donc cette valeur qui va être la valeur de repos de la variable L !

```
5 LET L=1
```

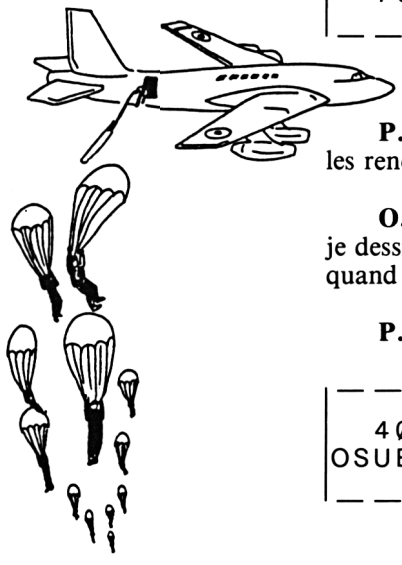


O. — Maintenant, je comprends tout !... Pour amorcer la descente, il suffit de donner à L la valeur 2 en écrivant :

```
40 IF INKEY$="Ø" THEN GOSUB 11
Ø
110 LET C=X
120 LET L=2
130 RETURN
```

O. — ... et en précisant devant chaque ligne de descente du parachute :

```
50 IF L>2 THEN PRINT AT L,C;"
";AT L+1,C;" "
60 IF L>1 THEN LET L=L+1
70 IF L>1 THEN PRINT AT L,C;"O
";AT L+1,C;"X"
75 IF L=19 THEN LET L=1
```



P. — Effectivement, tu as tout compris !... En écrivant ainsi les lignes 50 à 80, tu ne les rends opérantes que lorsque L est plus grand que 1 !... Bravo !...

O. — N'empêche que ce n'est pas parfait !... Si je reste appuyé sur la touche [0], je dessine toute une ribambelle de parachutistes dont seul le dernier veut bien descendre quand je n'appuie plus !... Il n'y a pas moyen de supprimer ce défaut ?

P. — Si, bien sûr !... Rajoute donc ceci à ta ligne 40 :

```
40 IF INKEY$="0 AND L=1 THEN G
OSUB 110
```

O. — Ça y est, ça marche !... C'est astucieux cette façon de faire !... Ainsi, tu ne peux donner la valeur 2 à L qu'une seule fois !

P. — Eh, oui !... C'est cela l'astuce !

O. — Tiens, en parlant d'astuce, il me vient une idée !

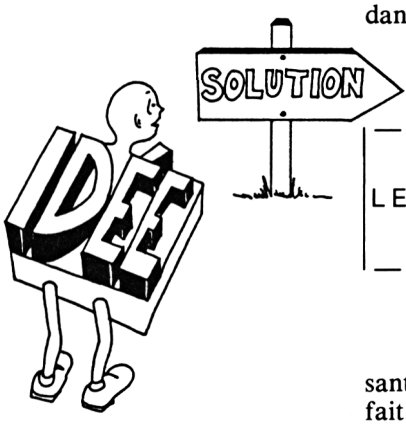
P. — Aïe !

O. — Merci !... Enfin, je te la dis quand même !... Il s'agit de la façon dont on attribue la valeur 2 à L et X à C en appuyant sur la touche [0]. Je crois que je viens d'imaginer une troisième méthode qui ne nécessite pas l'emploi d'un sous-programme !

P. — Intéressant !... Je t'écoute !

O. — Si, quand tu écris la ligne IF INKEY$ = ..., tu t'en sers pour attribuer d'abord la valeur 2 à L, tu peux ensuite te servir du fait que L vaut 2 pour affecter X à C !

P. — Il me semble que ton idée devrait marcher !... Ecris donc les lignes correspondantes pour vérifier !



O. — Tout de suite ... Voilà ce que ça donne :

```
40 IF INKEY$="0" AND L=1 THEN
LET L=2
45 IF L=2 THEN LET C=X
```

P. — Cela fonctionne !... Mes compliments !... Ta solution est drôlement intéressante, car elle permet d'économiser deux lignes d'instruction !... Bravo encore !... Au fait, tu ne m'as pas dit si le jeu te plaisait !

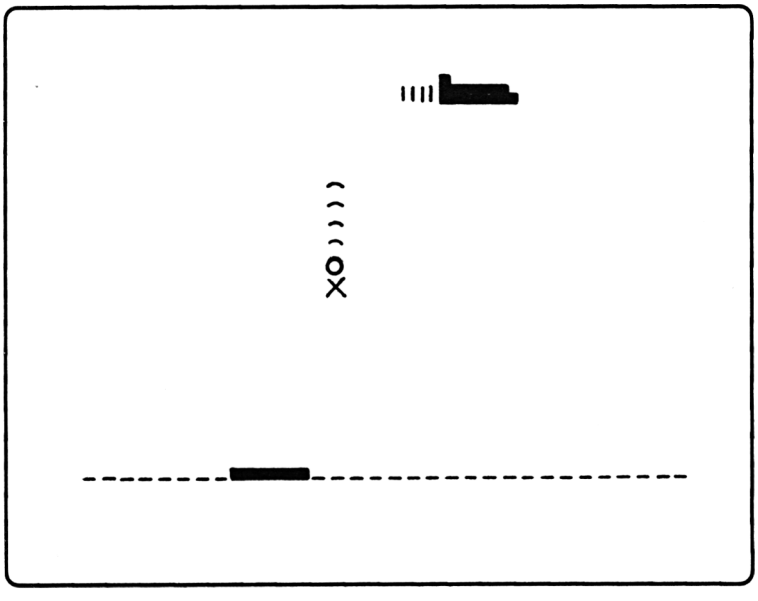
O. — Ouais, il n'est pas mal !... Mais encore une fois, c'est trop facile !... Oh, je viens d'avoir une autre idée !... Si on rajoutait du vent qui souffle en rafale ?

P. — Très bonne idée !... Tu veux essayer ?

O. — D'accord !... Voici donc un petit vent qui devrait souffler en rafale si je ne me suis pas trompé :

```
  1 LET C=Ø
 65 LET C=C-INT (RND*2)
```

P. — Bravo !... Décidément, je ne vais plus avoir grand chose à t'apprendre bientôt !



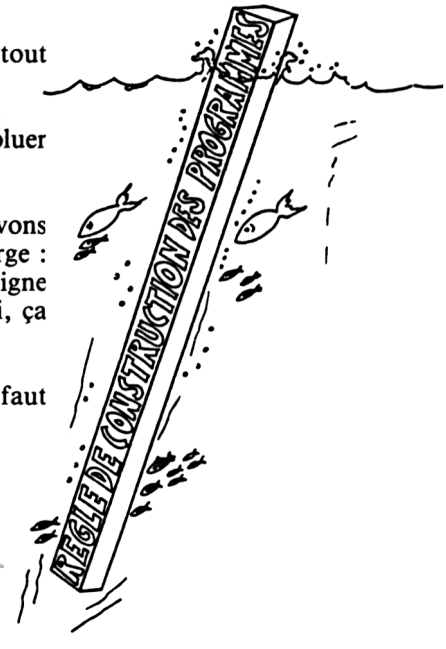
O. — Mais si, mais si !... As-tu remarqué que les programmes que nous écrivons depuis quelque temps deviennent de plus en plus longs ?

P. — C'est fatal !... Chaque nouvelle étape que je te fais franchir nécessite pratiquement l'emploi préalable de tout ce que tu as appris depuis le début !... Et ce n'est pas fini !

O. — Ça, je m'en serais douté !... En tout cas, je dois dire que jusqu'à présent tout va bien !

P. — J'avais remarqué !... Tu n'éprouves plus la moindre difficulté à faire évoluer un engin sur l'écran, malgré tous les problèmes d'effacement que cela pose !

O. — C'est comme pour les commandes au clavier !... D'après ce que nous avons écrit jusqu'à présent, il me semble qu'une règle de construction des programmes émerge : il faut que les lignes qui comportent les instructions INKEY $ soient placées entre la ligne d'effacement du mobile qu'elles commandent et celle de sa réimpression !... A toi, ça peut paraître évident, mais à moi, ça ne l'est que depuis peu !



P. — Ta remarque est tout à fait judicieuse !... C'est effectivement ainsi qu'il faut procéder !

O. — Bien !... Ceci dit, qu'est-ce qu'on fait maintenant ?

P. — Que penserais-tu d'étudier quelques petits programmes de tir !

O. — Super bonne idée !... Par lequel on commence ?

P. — Par le tir à partir d'un engin immobile !

La commande de tir

O. — J'imagine que ça doit être bougrement compliqué quand l'engin se déplace ?

P. — Un peu plus !... C'est pourquoi il vaut mieux débuter par le plus simple !

O. — Tu as une idée de ce que nous pourrions choisir comme engin ?

P. — Non !... Simplement, je te demande de ne pas employer d'engins de guerre pour cela !... Tu disposes de suffisamment d'imagination pour trouver une application de paix à ce type de programme.

O. — J'ai trouvé !... Je vais dessiner une fusée qui nettoie l'espace à coups de canon laser de tous les corps errants qui s'y promènent !

P. — Une sorte d'aspirateur cosmique, si je comprends bien !

O. — C'est ça !

P. — Alors, il ne te reste plus qu'à le dessiner, ton aspirateur !... Allez, au boulot !

O. — Attends, je réfléchis !... Je crois que je vais pouvoir perfectionner la chose !... Je vais dessiner une flotille de fusées et ce sera celle de tête qui jouera le rôle de nettoyeur d'astéroïdes !

P. — Excellente idée !... Ainsi, le jeu sera plus vivant !

O. — Voici donc la flotille de fusées :

```
10 FOR A=6 TO 14 STEP 2
20 PRINT AT A,0;"▟▙";AT A+1,0;
"▜▛"
30 NEXT A
```

P. — Elles sont très bien tes fusées !

O. — N'est-ce pas ?... A la fusée de tête maintenant :

```
 5 LET L=10
70 PRINT AT L,4;"▟▙";AT L+1,4;
"▜▛"
```

P. — Tiens, pourquoi avoir numéroté ta ligne 70 ?

O. — Parce que, maintenant, je sais que si je veux faire monter et descendre cette fusée de tête, il faut que je place avant la ligne que je viens d'écrire les deux lignes de commande INKEY$, elles-mêmes précédées de la ligne d'effacement !... Comme ça, quoi !

```
40 PRINT AT L,4;"  ";AT L+1,4;
"  "
50 IF INKEY$="6" AND L<14 THEN
LET L=L+1
60 IF INKEY$="7" AND L>6 THEN
LET L=L-1
```

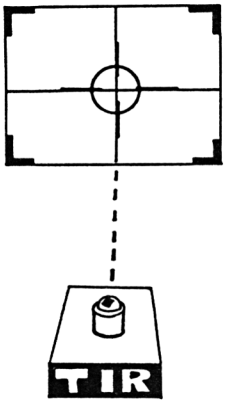
P. — ... sans oublier la ligne de bouclage, sinon cela ne fonctionne pas !

```
80 GOTO 40
```

O. — Bien sûr !... Et maintenant, comment on écrit le programme de tir ?

P. — On va déjà commencer par un petit programme tout simple qu'on améliorera par la suite :

```
 75 IF INKEY$="0" THEN GOSUB 10
0
100 FOR C=6 TO 32
110 IF C>6 THEN PRINT AT L,C-1;
" "
120 IF C<32 THEN PRINT AT L,C;"
•"
130 NEXT C
140 RETURN
```



O. — Je reconnais là un sous-programme qui déplace un projectile de la colonne 6 à la colonne 31, auquel on accède en appuyant sur la touche [0] !

P. — As-tu compris pourquoi le déplacement du projectile débutait à la colonne 6 ?

O. — Evidemment !... C'est parce que l'avant de la fusée occupe la colonne 5 !

P. — Bon, bon !... Alors, quel est ton avis sur ce programme de tir ?

O. — Pas mal au premier abord !... Toutefois, en y regardant de plus près, j'ai noté un défaut important !

P. — Lequel ?

O. — Quand je viens de tirer, c'est-à-dire quand je viens d'appuyer sur la touche 0, je suis obligé d'attendre que le projectile soit sorti de l'écran avant de pouvoir faire monter ou descendre la fusée !... Ça handicape énormément !

P. — Tu te doutes que cela est dû au fait qu'on fait appel à un sous-programme pour le tir !... Il faut donc attendre que le déroulement de ce sous-programme soit terminé pour revenir au programme principal dans lequel figurent les instructions de montée et de descente !

O. — Alors, puisque c'est ainsi, il ne faut pas faire appel à un sous-programme !

P. — Exact !... D'ailleurs, en règle générale, il ne faut jamais faire appel à un sous-programme lorsque l'on désire effectuer plusieurs opérations en même temps !

O. — Ta remarque vient de me faire penser à quelque chose !... Dans l'exemple précédent où l'on larguait le parachutiste à partir d'un avion en vol, on effectuait déjà deux opérations en même temps !... On doit pouvoir utiliser le même type de programme ?

P. — Pourquoi pas !... Veux-tu essayer ?

O. — Oui !... Je devrais y arriver !... Il faut d'abord que je repousse la ligne 80 car je vais placer les instructions de tir à la suite de la ligne 70 !... Ensuite, il faut que je reprenne la ligne 75 qui ne convient plus :

```
75 IF INKEY$="Ø" THEN LET X=6
8Ø IF X>6 THEN PRINT AT L,X-1;
" "
85 IF X=32 THEN LET X=5
9Ø IF X>5 THEN PRINT AT L,X;"•
"
94 IF X>5 THEN LET X=X+1
```

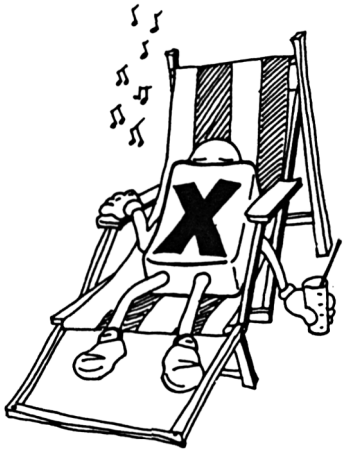
P. — Il me semble que tu as oublié deux petites choses indispensables !

O. — C'est vrai !... J'ai oublié de réécrire la ligne de fermeture de boucle :

```
1ØØ GOTO 4Ø
```

P. — ... et la ligne qui définit la valeur de repos de la variable X :

```
7 LET X=5
```



O. — Elle n'est pas utile cette ligne, puisque X est défini à la ligne 75 !

P. — Ah oui ?... Et que se passe-t-il si tu n'appuies pas sur la touche [Ø] ?

O. — J'obtiens un message d'erreur 2/8Ø !... J'ai compris !

P. — Ceci dit, tu regardes si ce programme convient mieux !

O. — Tiens, c'est drôle !... Maintenant que je peux faire monter ou descendre la fusée pendant que le projectile se déplace, je le fais monter ou descendre au même rythme que la fusée !

P. — Pour remédier à ce petit défaut, il suffit d'employer une variable différente de L comme coordonnée de ligne du projectile !

O. — Comme Y, par exemple ?

P. — Oui, très bien !... Il ne te reste plus alors qu'à initialiser cette variable Y avec la valeur que possédait L au moment où tu appuyais sur la touche [Ø] !

O. — Ah, comme tout à l'heure !... Attends, je vais employer pour cela « ma méthode à moi » :

```
78 IF X=6 THEN LET Y=L
```

O. — ... et je vais remplacer la variable L par Y dans les lignes 8Ø et 9Ø :

```
8Ø IF X>6 THEN PRINT AT Y,X-1;
" "
9Ø IF X>5 THEN PRINT AT Y,X;"•
"
```

P. — Et voilà, le tour est joué !... Es-tu content de ton nouveau programme ?

O. — Pas encore !... Quand je reste appuyé sur la touche [Ø], le projectile n'avance plus !... Pourquoi ?

P. — Tout simplement parce que tu maintiens la valeur de X à 6 !... Il te faut rajouter dans ta ligne 75 la même instruction que celle qu'on avait dû rajouter dans l'exercice précédent pour empêcher l'apparition de plusieurs parachutistes !... Tu essayes de trouver tout seul ?

O. — C'est déjà fait !... Voici la solution :

```
 75 IF INKEY$="Ø" AND X=5 THEN
LET X=6
```

P. — Parfait !... J'espère que ce coup-ci, tu es enfin satisfait !

O. — Tu vas dire que je suis difficile, mais il reste encore un point à régler !... Imagine, qu'après avoir tiré, je m'aperçoive que j'ai mal visé !... Il faut que je puisse retirer aussitôt après avoir monté ou descendu la fusée au bon niveau !... Or, dans l'état actuel du programme, je ne le peux pas !

P. — Cette petite difficulté ne me paraît pas insurmontable !... Je dirais même qu'après réflexion, la solution me paraît toute simple !... Il suffit de revenir à la situation précédente pour la ligne 75, c'est-à-dire qu'il faut enlever AND X = 5, et déplacer la ligne d'effacement du projectile pour la placer avant cette même ligne 75 :

```
 72 IF X>6 THEN PRINT AT Y,X-1;
" "
 75 IF INKEY$="Ø" THEN LET X=6
```

O. — C'est bon !... Seulement, je retrouve le défaut du projectile qui n'avance plus si je maintiens mon doigt appuyé sur la touche !

P. — Qu'est-ce que tu es compliqué ?... Il faudra pourtant t'en contenter ou bien trouver tout seul une solution car, moi, je suis à court d'idée !

O. — Bon !... Et bien, je chercherai plus tard !... En attendant, si on se dessinait quelques astéroïdes errant à travers l'espace ?

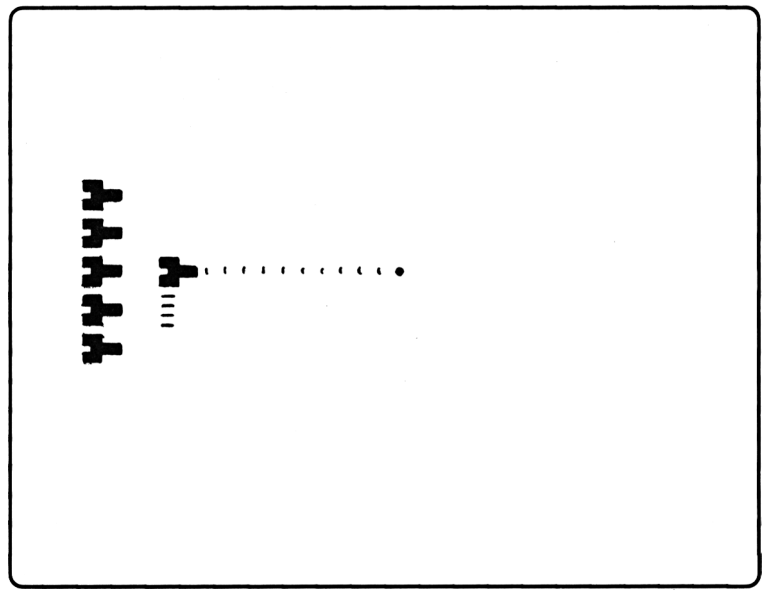
P. — Bonne idée !... Comme je suppose que le « si on dessinait » veut dire « si tu dessinais », je vais donc t'écrire le programme :

```
 95 PRINT AT A,B;" "
 96 IF B=3Ø THEN LET A=INT (RND
*9)+6
 97 LET B=B-1
 98 IF B=-1 THEN LET B=3Ø
 99 PRINT AT A,B;"*"
```

P. — ... sans oublier, comme je viens de le faire, la ligne qui détermine la valeur initiale de B :

```
 8 LET B=3Ø
```

O. — Ça y est, c'est parti !... Le temps que je m'aligne et feu !... Plus d'astéroïdes !... Tiens, c'est pas le cas !... Le projectile le traverse comme si de rien n'était !... Tu as une idée pourquoi ?

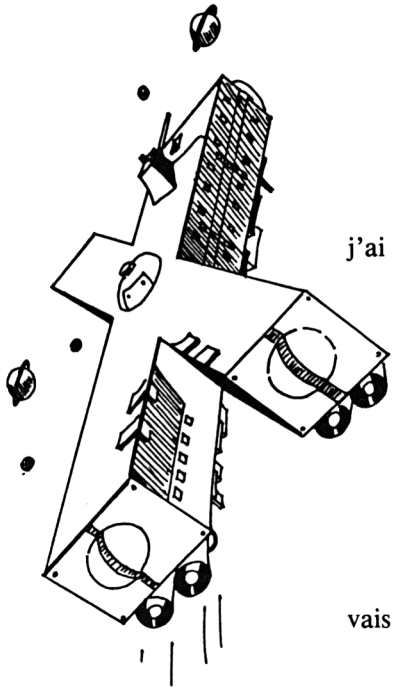


P. — Oh que oui !... Tel que nous avons écrit ce jeu, il ne pouvait pas en être autrement, car nous n'avons aucun programme d'impact !...

O. — Et je suppose que tu vas me dire que nous allons étudier les programmes d'impact dans huit jours ?

P. — Exactement !... Si tu veux bien attendre à samedi prochain, on pourra alors terminer ce jeu complètement !

O. — Ok, alors vivement samedi prochain !



Le tir laser à partir d'un véhicule en déplacement

P. — Pour te faire patienter jusque-là, je te propose d'étudier un sujet intéressant, j'ai nommé le tir laser !

O. — Intéressant ?... Passionnant tu veux dire !

P. — Puisque cela te plaît, je te laisse choisir le choix des armes !

O. — Bien !... Alors, je choisis un vaisseau de l'espace en forme de grand X !

P. — Et sur quoi va-t-il tirer, ton vaisseau de l'espace ?

O. — Sur tous les fragments de planète qui vont se présenter devant lui !

P. — Bonne idée !... Pour t'aider à rendre ce jeu plus captivant que le précédent, je vais te faire une suggestion !

O. — Je t'écoute !

P. — Dans le jeu précédent, ton engin était immobile et c'était les astéroïdes qui bougeaient. C'est d'ailleurs parce que ces derniers se déplaçaient qu'il te fallait un programme d'impact !

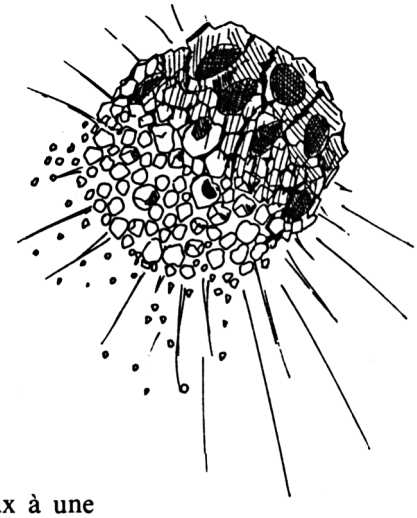
O. — Oui !... D'ailleurs, si j'ai bien compris, l'astéroïde touché ne s'effaçait pas parce que le programme le réimprimait systématiquement de façon à créer son déplacement !

P. — C'est tout à fait cela !... Or, ce processus peut être inversé, c'est-à-dire que tu peux rendre le vaisseau mobile et les fragments de planètes immobiles !

O. — Et dans ce cas, comme tu n'imprimes qu'une seule fois chaque fragment, il va être possible de l'effacer en lui tirant dessus, comme si on le désintégrait !

P. — Exact !... Alors à toi de jouer !

O. — C'est parti !... Voici déjà le vaisseau de l'espace :



```
   5 LET L=10
   6 LET C=0
  10 PRINT AT L,C;"▚▞";AT L+1,C;
"■■"
```

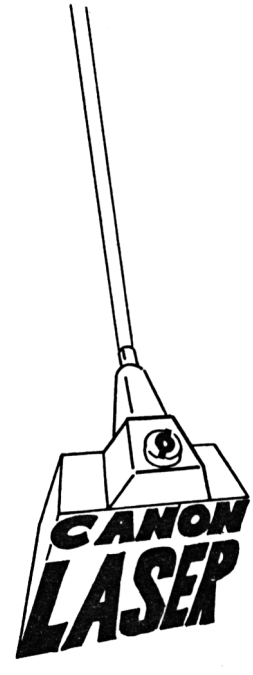
P. — Très beau !... On le fait monter et descendre, j'imagine ?

O. — Oui !... Par contre, j'ai envie de limiter ses déplacements verticaux à une dizaine de lignes !

P. — Excellente idée !... Cela devrait retarder l'apparition de la saturation de la mémoire d'affichage et, de surcroît, on aura l'impression de voler dans un couloir de l'espace !

O. — Bon, alors voici les lignes d'instructions nécessaires :

```
  10 PRINT AT L,C;"  ";AT L+1,C;
"  "
  20 IF INKEY$="6" AND L<14 THEN
 LET L=L+1
  30 IF INKEY$="7" AND L>6 THEN
LET L=L-1
  40 PRINT AT L,C;"▚▞";AT L+1,C;
"■■"
  50 GOTO 10
```



P. — C'est devenu de la routine, n'est-ce pas, que d'écrire ce genre de programme ?

O. — Ben oui !... C'est presque évident maintenant !

P. — C'est la preuve que tu fais des progrès !... Je suis persuadé que tu es capable de venir à bout du programme de tir laser tout seul !

O. — Je veux bien essayer, encore faudrait-il que je sache par quoi commencer !

O. — Tu vas commencer par dessiner un faisceau laser qui part du véhicule spatial quand on appuie sur la touche [0] !

O. — Ça, c'est facile !... Voici la chose :

```
   9 LET X=1
  50 IF INKEY$="Ø" THEN LET X=X+
1
  60 PRINT AT L,X;"-"
  70 GOTO 10
```

P. — Je vois que tu as décidé d'employer le signe moins pour dessiner le faisceau laser !

O. — C'est ce qui m'a semblé le plus réaliste !

P. — Bien !... Ce que je trouve bien aussi, c'est le fait que la longueur du faisceau dépend du temps pendant lequel on appuie sur la touche [Ø] !

O. — Pour tout te dire, je ne l'ai pas fait exprès !... C'est en prime que je l'offre !

P. — Alors, il faut empêcher le programme de se bloquer si on reste trop longtemps appuyé sur la touche de commande !

O. — Je vois ce que c'est !... Il suffit de rajouter à la ligne 5Ø :

VITESSE D'UN RAYON LUMINEUX

```
  50 IF INKEY$="Ø" AND X<31 THEN
 LET X=X+1
```

MAX. 300000Km/s

P. — Ok, c'est bon !... Toutefois, si tu me permets une suggestion, je trouve que le faisceau laser ne progresse pas assez vite ! Comme il s'agit d'un faisceau de lumière, il faudrait que tu trouves une solution pour obtenir une progression plus rapide !

O. — Je sais comment je vais faire !... Je vais commander le déplacement de deux carreaux à chaque fois !... Ainsi, cela va doubler sa vitesse !

P. — Bonne idée !... J'ajouterai même que si tu trouves encore la vitesse insuffisante, tu peux essayer plus de deux carreaux !

Minimum

O. — Ok, je note !... En attendant, voici comment j'obtiens une progression de deux en deux en modifiant les lignes 5Ø et 6Ø :

```
  50 IF INKEY$="Ø" AND X<31 THEN
 LET X=X+2
  60 PRINT AT L,X;"--"
```

P. — La modification que tu viens d'exécuter fait apparaître un défaut !

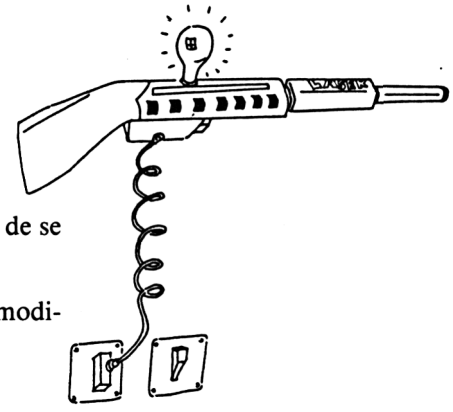
O. — Je l'avais remarqué !... L'un des pointillés s'imprime en permanence à la droite du vaisseau spatial !... Il va donc falloir que je modifie à nouveau la ligne 6Ø pour empêcher cela :

```
  60 IF X>1 THEN PRINT AT L,X;"-
-"
```

P. — Oui, mais maintenant, il manque le début du faisceau lorsque l'on appuie sur la commande de tir !

O. — Décidément, je n'en sortirai pas !... Il faut que je décale d'une colonne vers la gauche la valeur initiale de X !... Cela donne :

```
  9 LET X=Ø
 6Ø IF X>Ø THEN PRINT AT L,X;"-
-"
```



P. — Cette fois, c'est bon !... Il ne te reste plus qu'à empêcher le programme de se bloquer lorsque le faisceau atteint le bord droit de l'écran et le tour sera joué !

O. — C'est vrai que quand j'ai augmenté le pas de progression, j'ai oublié de modifier la valeur limite !... Voilà, l'oubli est réparé :

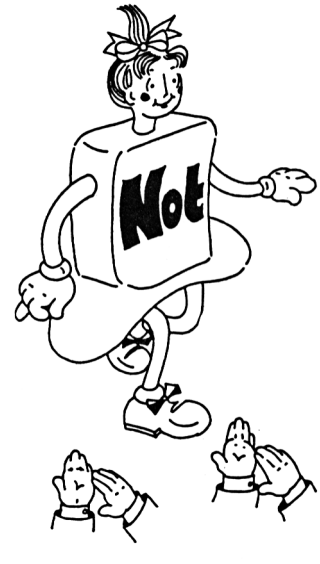
```
 5Ø IF INKEY$="Ø" AND X<3Ø THEN
LET X=X+2
```

P. — Bien !... Maintenant, il faut éteindre le faisceau dès qu'on n'appuie plus sur la touche de commande. Pour cela, je te propose la méthode suivante qui consiste à lancer une nouvelle variable chargée d'assurer l'effacement du faisceau sur toute sa longueur :

```
 7Ø IF NOT INKEY$="Ø" AND X>Ø T
HEN LET F=F+1
 8Ø IF F>Ø THEN PRINT AT L,F;"
"
 9Ø GOTO 1Ø
```

P. — ... sans oublier d'initialiser la nouvelle variable F comme je viens de le faire :

```
 8 LET F=Ø
```



O. — Tiens, une petite nouvelle !

P. — Eh oui !... Que veux-tu, il en faut bien de temps en temps !... Celle-ci s'appelle NOT. C'est une instruction chargée de définir le contraire d'une condition.

O. — Je crois que je viens de comprendre !... Dans notre exemple, IF NOT INKEY$="Ø" veut dire « si on n'appuie pas sur la touche [Ø] » !

P. — Tout à fait exact !... Je vois que tu as bien saisi l'emploi de cette nouvelle instruction !

O. — En attendant, nouvelle instruction ou pas, l'effacement du faisceau est trop lent et n'est pas total !

P. — Attends, on va arranger cela en effaçant deux carreaux à la fois, en modifiant les lignes 7Ø et 8Ø ainsi :

```
  70 IF NOT INKEY$="0" AND X>0 T
HEN LET F=F+2
  80 IF F>0 THEN PRINT AT L,F;"
 "
```

O. — Ça va mieux, mais il reste deux pointillés qui ne s'effacent pas !

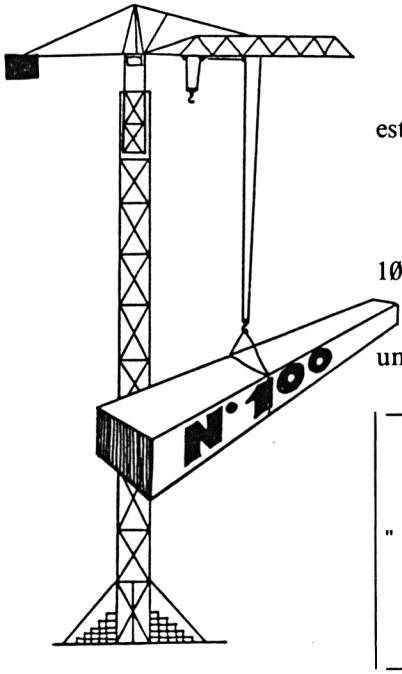
P. — En réalité, la ligne 80 les efface bien, mais c'est la ligne 60 qui les réimprime systématiquement. Ce qu'il manque, c'est une ligne de détection de la fin de l'effacement !

O. — Quelque chose comme :

```
  90 IF F=X THEN LET X=0
```

P. — Oui, à condition de la faire suivre par une ligne de remise à zéro de la variable F :

```
 100 IF X=0 THEN LET F=0
 110 GOTO 10
```



O. — C'est bon, ça marche !

P. — Puisque cela marche, je te propose d'analyser en détail ton programme, car il est possible d'économiser une ligne d'instruction qui se répète deux fois !

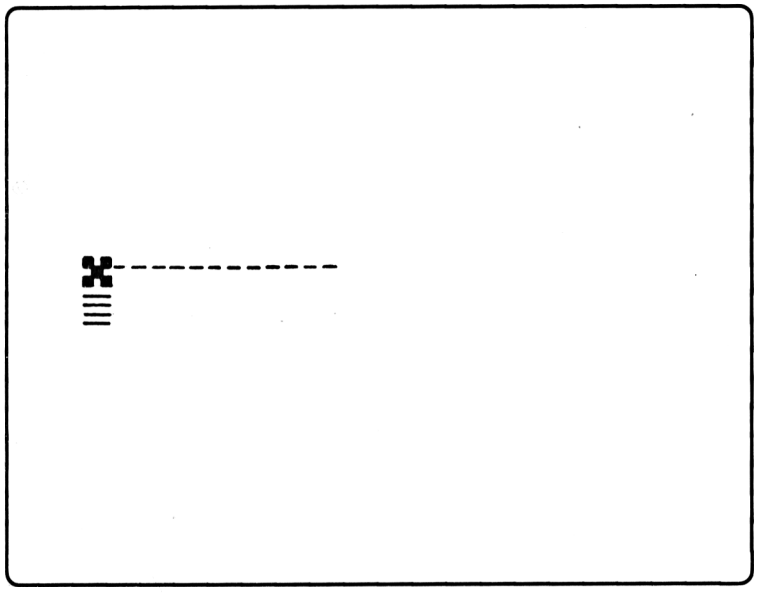
O. — Ah bon !... Laisse-moi chercher ça !... Non, je ne vois pas !

P. — Il s'agit de la ligne 8 que tu peux supprimer à condition de remonter la ligne 100 à la place de la ligne 10 !

O. — Je viens de comprendre !... Il est effectivement possible de définir F = 0 avec une seule ligne en opérant ainsi :

```
   8
  10 IF X=0 THEN LET F=0
  11 PRINT AT L,C;"   ";AT L+1,C;
"   "
 100
```

P. — Bien, voilà une bonne chose de faite !... Il ne reste plus qu'à réaliser la même chose à partir d'un véhicule spatial en déplacement !



O. — Voici déjà de quoi faire avancer le vaisseau :

```
35 LET C=C+1
37 IF C=31 THEN LET C=Ø
```

P. — Très bien !... Cependant, comme tu peux t'en rendre compte, le programme de tir ne convient plus !

O. — Ça, tu peux le dire !... Je vais d'ailleurs l'effacer entièrement !... Comme ça, je m'y retrouverai mieux !

P. — Comme tu voudras !... Je peux même te proposer une autre formule de tir laser, si tu veux !

O. — Tu penses si je veux !... Comment il est ce nouveau tir laser ?

P. — Il s'agit de créer une salve de trois carreaux qui, une fois émise, se propage toute seule à une vitesse triple de celle du vaisseau !... Cela te convient ?

O. — Super !... Voici déjà de quoi commander le tir :

```
50 IF INKEY$="Ø" AND X=Ø THEN
LET X=C+2
```

P. — Explique-moi pourquoi tu donnes à X la valeur C + 2 !

O. — Ça, c'est parce que le vaisseau occupe deux colonnes, C et C + 1, ce qui fait que le faisceau ne peut débuter qu'à partir de la colonne C + 2 !

P. — Ok !... Continue !

O. — Voici donc les lignes d'impression et de progression du faisceau :

```
60 IF X>0 THEN PRINT AT L,X;"-
--"
70 IF X>0 THEN LET X=X+3
```

O. — ... suivies de la ligne de sortie d'écran du faisceau :

```
80 IF X>29 THEN LET X=0
```

P. — Bien !... Il ne te reste plus qu'à écrire la ligne d'effacement du faisceau !

O. — Ça, c'est moins évident !... J'ai bien pensé à écrire :

```
55 IF X>0 THEN PRINT AT L,X-3;
"  "
```

O. — ... mais comme tu peux t'en rendre compte, ça ne marche pas !... Le vaisseau s'efface au début du tir et le faisceau n'est pas effacé en sortie d'écran !

P. — Attends, je vais t'aider !... Pour effacer le faisceau en sortie d'écran, il te faut modifier ta ligne ainsi :

```
55 IF X>0 THEN PRINT AT L,X;"
"
```

P. — ... et déplacer ta ligne 60 en 90, de façon que la ligne 70 de progression soit située entre la ligne d'effacement et la ligne d'impression :

```
60
90 IF X>0 THEN PRINT AT L,X;"-
--"
```

O. — Ça marche pour l'effacement en sortie d'écran !... Par contre, le faisceau ne débute que trois carreaux après le vaisseau !... Comment remédier à cela ?

P. — On a déjà fait cela !... Il suffit d'interdire l'effacement sur les trois cases qui suivent le vaisseau, en écrivant :

```
55 IF X>C+2 THEN PRINT AT L,X;
"  "
```

O. — Tu sais, je ne voudrais pas te décevoir, mais cela ne change rien !

P. — Ah !... Attends que je réfléchisse !... J'ai trouvé !... C'est dû au fait que nous avons déplacé la ligne d'impression du faisceau !... Si tu examines le programme, tu vas t'apercevoir qu'à la ligne 5Ø on détermine une valeur de X égale à C + 2 et qu'avant d'avoir eu le temps d'imprimer une salve de trois signes moins, on augmente X de trois !

O. — C'est vrai ça !... Dans ces conditions, la solution est toute simple !... Il suffit de déterminer ainsi la valeur initiale de X :

```
50 IF INKEY$="Ø" AND X=Ø THEN
LET X=C-1
```

O. — ... et de remodifier la ligne 55 en tenant compte de cette nouvelle valeur :

```
55 IF X>C-1 THEN PRINT AT L,X;
"    "
```

P. — On y est presque !... Il ne te reste plus qu'à résoudre un petit problème de clignotement du vaisseau dû à l'effacement du faisceau et à empêcher le tir de se bloquer si l'on appuie en permanence sur la touche [Ø] !

O. — Pour le clignotement, je sais d'où ça provient !... C'est lié au fait que le vaisseau a avancé d'un carreau au moment où s'est fait l'effacement de la première portion du faisceau !

P. — Alors, je te suggère de placer la ligne d'effacement du faisceau avant celle d'impression du vaisseau spatial !

O. — Bonne idée !... Je vais la numéroter 38 :

```
38 IF X>C-1 THEN PRINT AT L,X;
"    "
55
```

P. — Quant au blocage du tir, j'ai découvert d'où cela provenait !... Quand on maintient le doigt appuyé sur la touche [Ø], on donne à X la valeur − 1 lorsque C vaut Ø. Et comme le tir ne se déclenche que si X=Ø...

O. — Ah, c'est ça !... Alors, il suffit de modifier ainsi la ligne 5Ø :

```
50 IF INKEY$="Ø" AND X<1 THEN
LET X=C-1
```

P. — Eh bien, tu vois qu'on y est arrivé !... Il ne nous reste plus qu'à empêcher la ligne de tir de suivre les variations d'altitude du vaisseau spatial et à dessiner quelques petits bouts de planète qu'il faudra désintégrer au passage.

O. — Pour la ligne de tir, je sais faire !... Il suffit de poser une variable d'altitude propre au faisceau et à définir sa valeur chaque fois qu'on appuie sur la touche de commande du tir !... Quelque chose comme :

```
60 IF X=C-1 THEN LET Y=L
```

P. — Pas mal !... N'oublie pas de modifier en conséquence les lignes 38 et 9Ø !

O. — J'allais le faire !... Voilà, c'est fait :

```
38 IF X>C-1 THEN PRINT AT Y,X;
"  "
9Ø IF X>Ø THEN PRINT AT Y,X;"-
--"
```

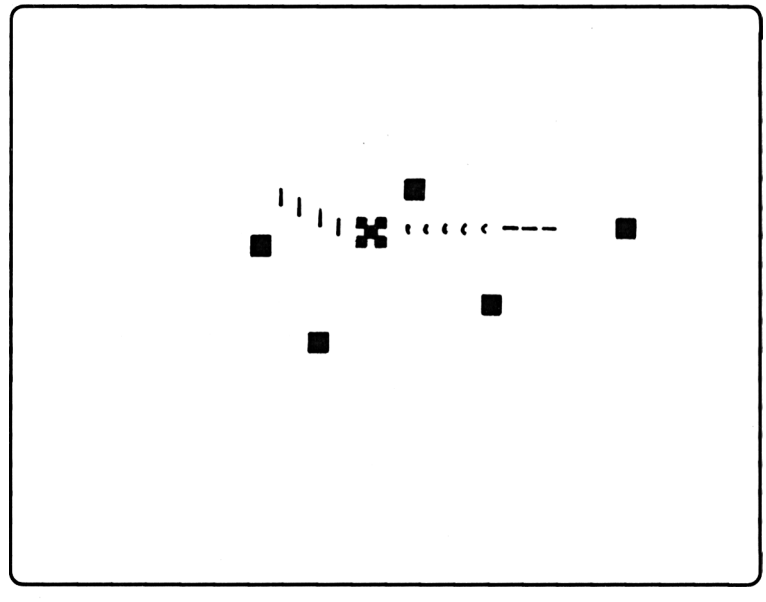
P. — Bien !... Pour les fragments de planète, je te suggère de les faire apparaître chaque fois que C est égal à zéro !

O. — Trop facile !... Il en faut au moins deux à chaque nouveau passage !

P. — Comme tu voudras !... Voici donc les obstacles :

```
1ØØ IF C=Ø OR C=16 THEN PRINT A
T INT (RND*9)+6,INT (RND*22)+8;"
*"
```

O. — Extra super !... Ce jeu est le meilleur qu'on ait réalisé jusqu'à présent !



Et c'est sur cette bonne parole d'Olivier, récompense de tant d'heures passées ensemble, que je décidais de clore cette journée qui, de toute façon, touchait déjà à sa fin. J'étais en train de mettre de l'ordre dans mes notes quand, du haut du grenier, Olivier m'annonça :

O. — Devine ce qui se passe ?... La mémoire est à nouveau saturée !... J'en ai assez !... Je ne peux arriver à jouer tranquillement sans qu'elle ne me rappelle sa présence, celle-là !... Dis Papa, il faut absolument que, pour la semaine prochaine, tu m'achètes la mémoire 16 K !... Ça ne peut plus continuer ainsi !

P. — D'accord, cela me paraît devenir indispensable !

REBONDS ET IMPACTS

Et c'est ainsi que, le vendredi suivant, je rentrai à la maison avec un petit paquet sous le bras. Olivier, toujours là où il ne faut pas, m'attendait l'air innocent sur le pas de la porte. Je ne saurais jamais si cette prémonition des événements était due à sa très forte intuition ou bien à l'existence d'oreilles de cocker qui traînent un peu partout et spécialement dans la cuisine lorsque, ce matin-là, je prévenai que je prenais le chéquier pour effectuer un achat important. En tout cas, il était là et bien là !... Pour l'effet de surprise, c'était raté !... J'annonçai tout de même :

P. — Olivier, j'ai une surprise pour toi !

O. — Je parie que c'est la mémoire 16 K ?

P. — Tu as gagné !

O. — Youpi !... Je vais enfin pouvoir aller au bout de mes programmes !... Merci Papa, tu es chic !

Ce compliment m'alla droit au cœur !... Entre temps, Olivier venait de défaire le paquet et en sortait la mémoire 16 K.

O. — Dis donc, c'est tout petit ce « machin »-là !... C'est pas croyable qu'il puisse y avoir 16 000 emplacements de mémoire là-dedans !

P. — Eh si !... On fait même encore plus petit, mais pour le moment, c'est encore trop cher !

O. — Ah bon !... Eh bé, chapeau pour la technique !... Tu me montres comment on la branche ?

P. — Pour cela, il faut l'enficher à l'arrière du micro-ordinateur, à l'endroit où il y a une découpe prévue à cet effet !

O. — Ça y est, c'est fait !

P. — Bien !... Dorénavant, il va falloir laisser cette mémoire branchée en permanence. Le circuit imprimé du micro-ordinateur qui reçoit le connecteur de la mémoire est trop fragile pour supporter un nombre important d'enfichages et de désenfichages. Si tu veux éviter que n'apparaissent les inévitables faux contacts qui vident en un clin d'œil la mémoire, je te conseille de ne plus y toucher une fois qu'elle est en place.

O. — Donc, si je comprends bien, nous allons désormais travailler avec cette mémoire 16 K !... Et que se serait-il passé si tu ne me l'avais pas achetée ?

P. — Nous n'aurions pas pu aller beaucoup plus loin !... Tout au plus, nous aurions pu étudier les rebonds, à conditions de travailler sur une faible portion de l'écran !... Quant à la suite, pas question d'étudier le moindre programme d'impact et encore moins le dialogue avec la machine !

O. — Ça aurait été bien dommage !

Et c'est sur cette dernière remarque, pleine de promesses pour l'avenir, que se clôtura la discussion sur la mémoire 16 K.

Le lendemain matin, je n'avais pas encore mis le pied par terre que j'entendais Olivier travailler au grenier. Je lui lançai :

P. — Qu'est-ce que tu fais de beau ?

O. — J'essaye les rebonds, mais ce n'est pas évident !

P. — Ok !... Laisse-moi déjeuner et j'arrive !

Ce qui fut dit fut fait !... Une demi-heure après, je prenais place à côté de lui. Je lui demandai :

P. — Alors, où est-ce que tu coinces ?

O. — Un peu partout !

P. — Je vois ce que c'est !... Si tu veux bien, on va étudier les rebonds avec méthode !... Pour commencer, voyons en détail les rebonds à l'horizontale !

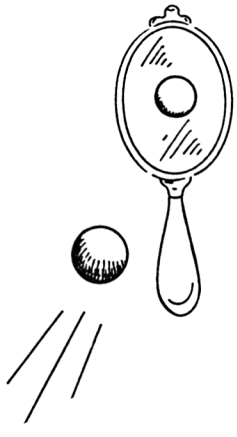
Les rebonds à l'horizontale

P. — Je ne sais si tu te souviens, mais le bord gauche de l'écran se comporte comme un miroir !

O. — Si, si, je m'en souviens !... Mais quel rapport cela a-t-il avec les rebonds ?

P. — Ecris donc le programme suivant, et dis-moi ce qui se passe :

```
 5 LET C=16
10 PRINT AT 10,C;"  "
20 LET C=C-1
30 PRINT AT 10,C;"O"
40 GOTO 10
```



O. — Tiens, c'est intéressant !... La balle se déplace de droite à gauche, et quand elle atteint le bord gauche de l'écran, elle rebondit et retraverse l'écran pour aller se perdre sur le bord droit !... Pourquoi rebondit-elle ainsi ?

P. — A cause de l'effet de miroir que présente le bord gauche de l'écran !... Si tu te souviens bien ce qui se passe dans ce cas-là, l'affichage ne tient pas compte du signe moins de la coordonnée de colonne. Ainsi, lorsque apparaît la valeur -1 pour la variable C, l'affichage se fait dans la colonne $+1$, ce qui donne l'illusion du rebond !

O. — Ah, c'est ça !... C'est une sorte de rebond automatique, quoi ?

P. — Exact !... Par contre, le bord droit de l'écran ne se comporte pas de la même façon !... Si tu veux obtenir un rebond sur ce bord-là, il faut écrire un véritable programme de rebond, comme celui-ci :

```
 1 LET U=1
 5 LET C=16
10 PRINT AT 10,C;"  "
20 LET C=C+U
30 PRINT AT 10,C;"O"
40 IF C=31 THEN LET U=-1
50 GOTO 10
```

O. — En fin de compte, ce programme de rebond ne diffère d'un programme de déplacement que par les lignes 2Ø et 4Ø !

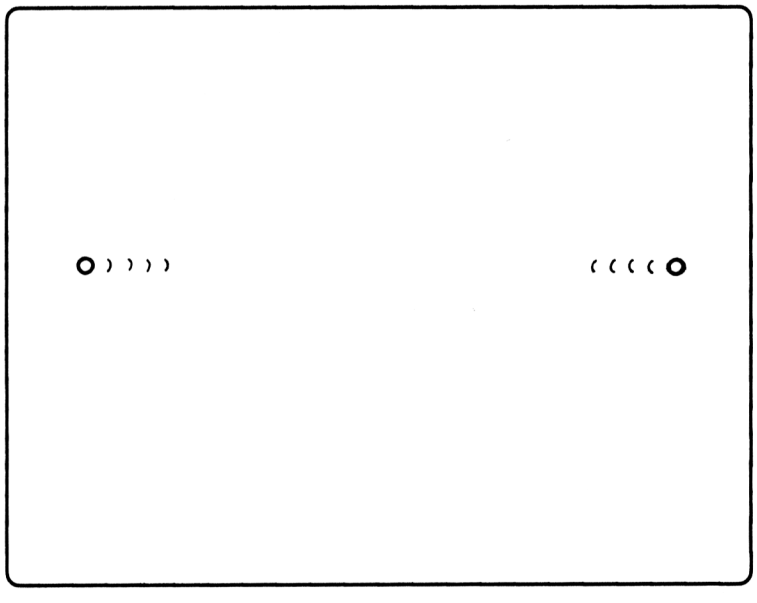
P. — Exactement !... A la ligne 2Ø, tu ajoutes à C une quantité U qui vaut tantôt 1, tantôt −1. Lorsque U vaut 1, tu obtiens un déplacement de gauche à droite, qui s'inverse lorsque U vaut −1.

O. — C'est génial comme « truc » !... Il suffit alors de changer la valeur du signe de U lorsque la balle arrive sur la colonne 31 pour obtenir le rebond sur le côté droit de l'écran !

P. — C'est ce que fait la ligne 4Ø !... Maintenant, je voudrais ton avis sur le fonctionnement de ce programme !

O. — Attends, je l'essaye !... Tiens, c'est bizarre !... Après un rebond à droite, suivi d'un rebond à gauche, la balle se perd à droite et l'ordinateur me délivre un message d'erreur B/3Ø. Pourquoi ?

P. — Parce que, après le rebond à droite, la valeur de U est négative, ce qui diminue alors la valeur de C de 31 à Ø, puis de Ø à −31 après le rebond automatique sur le bord gauche !... Comme ensuite tu cherches à détecter le moment où C vaut 31 et que tu ne trouves rien, puisque C vaut −31, il n'y a plus rien pour empêcher l'ordinateur de vouloir imprimer la balle dans la colonne suivante qui, bien sûr, n'existe pas !



O. — Alors, que dois-je faire ?

P. — Normaliser le rebond sur le bord gauche de l'écran à l'aide de la ligne suivante :

```
45 IF C=Ø THEN LET U=1
```

O. — Ça y est, c'est bon !... Dans le fond, c'est tout simple !... A moi les rebonds à la verticale !

Les rebonds à la verticale

P. — Excellente idée !... Comme le haut de l'écran se comporte comme le côté gauche, cela va représenter pour toi un excellent exercice d'application de ce que je viens de t'apprendre !

O. — Voici donc le rebond automatique sur le haut de l'écran :

```
 5 LET L=10
10 PRINT AT L,16;"  "
20 LET L=L-1
30 PRINT AT L,16;"O"
40 GOTO 10
```

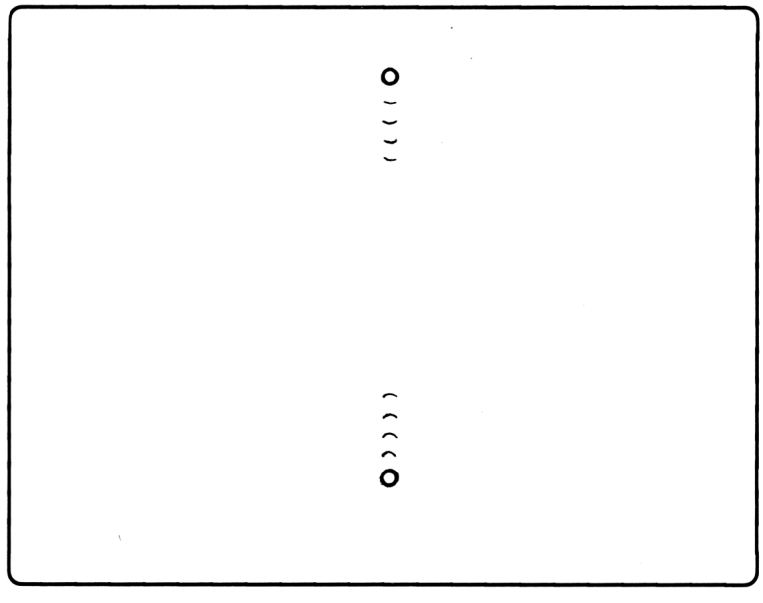
O. — ... suivi du rebond sur le bas de l'écran :

```
 1 LET V=1
 5 LET L=10
10 PRINT AT L,16;"  "
20 LET L=L+V
30 PRINT AT L,16;"O"
40 IF L=21 THEN LET V=-1
50 GOTO 10
```

O. — ... avec, pour terminer, la ligne qui remplace le rebond automatique sur le haut de l'écran :

```
45 IF L=0 THEN LET V=1
```

P. — J'espère que cet exercice ne t'a pas trop fatigué ?



O. — Non, ça va bien, merci !... Ce n'est quand même pas de ma faute s'il suffisait de remplacer la coordonnée de colonne par celle de ligne pour obtenir ce programme de rebond à la verticale !

P. — En tout cas, je suis content de voir que cela te paraît simple !... Puisque tu es dans de si bonnes dispositions, serais-tu capable d'accélérer la vitesse de progression de la balle ?

O. — Facile !... Il suffit de progresser de deux en deux et pour cela, il faut modifier les lignes 1, 4Ø et 45 ainsi :

```
 1 LET V=2
40 IF L=20 THEN LET V=-2
45 IF L=0 THEN LET V=2
```

P. — Je m'incline !... Tu as même pensé à modifier la valeur de L qui effectue le changement de sens !

O. — Eh oui !... Il fallait la rendre paire puisque la balle progressait de deux carreaux à chaque bond !

P. — Comme je ne vois plus rien à te dire sur les rebonds à la verticale, je te suggère de passer à ceux en oblique !

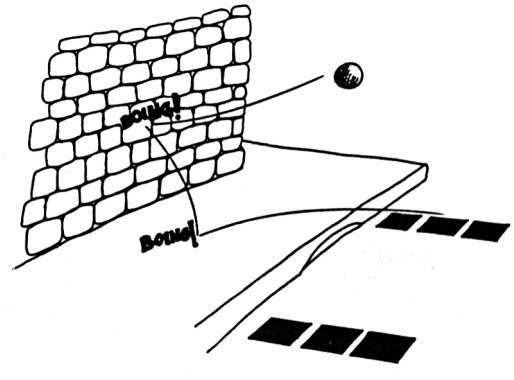
Les rebonds en oblique

O. — Je n'ai pas l'impression que ça va beaucoup changer !

P. — Tu ne te trompes pas !... Je vais toutefois t'écrire les divers programmes de rebond en oblique sur les quatre côtés de l'écran et je te laisserai le soin de trouver un programme unique qui les assure tous les quatre !

O. — D'accord comme ça !

P. — Alors, voici le premier rebond sur le côté gauche de l'écran :




```
 1 LET U=-1
 2 LET V=-1
 5 LET C=5
 6 LET L=21
10 PRINT AT L,C;"  "
20 LET C=C+U
30 LET L=L+V
40 PRINT AT L,C;"O"
50 IF C=0 THEN LET U=1
60 GOTO 10
```

P. — Tu noteras que, désormais, j'évite tout rebond automatique !

O. — J'avais remarqué !... J'ai noté aussi qu'il suffisait de ne changer la valeur que d'une seule variable sur les deux pour obtenir un rebond en oblique !

P. — Bien vu !... Dans le cas présent, seule la valeur de U passe de − 1 à + 1 lors du rebond, puisque la balle continue de grimper !

O. — Pour le deuxième rebond sur le haut de l'écran, tu ne vas pas avoir grand-chose à écrire puisqu'il fonctionne déjà !

P. — Je vais toutefois empêcher le rebond automatique de venir perturber les futurs rebonds sur le bas et le côté droit de l'écran, en écrivant :

```
52 IF L=Ø THEN LET V=1
```

O. — Dis donc, tu m'as tendu un piège au début !... Pour obtenir un programme qui assure les rebonds sur les quatre côtés, il suffit de rajouter au programme existant deux nouvelles lignes, une pour le rebond sur le côté droit et l'autre pour celui sur le bas de l'écran !

P. — Je voulais que tu te rendes compte que c'est tout simple d'écrire ce genre de programme !... Veux-tu le finir ?

O. — Tout de suite !... Voici la ligne qui va assurer le rebond sur la droite :

```
54 IF C=31 THEN LET U=-1
```

O. — ... et voici celle du rebond vers le bas :

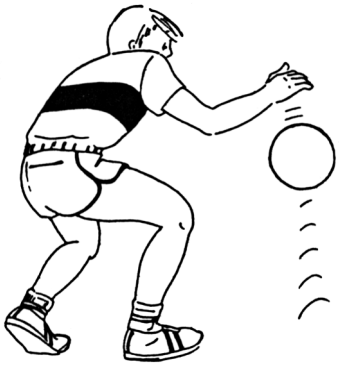
```
56 IF L=21 THEN LET V=-1
```

P. — Bien joué !... Voilà ce qui termine les rebonds en oblique sur les quatre côtés de l'écran !

O. — Je viens d'avoir une idée !... Je vais dessiner un terrain de jeux et je vais faire rebondir la balle à l'intérieur !

P. — Très bonne idée !... Je te laisse faire !

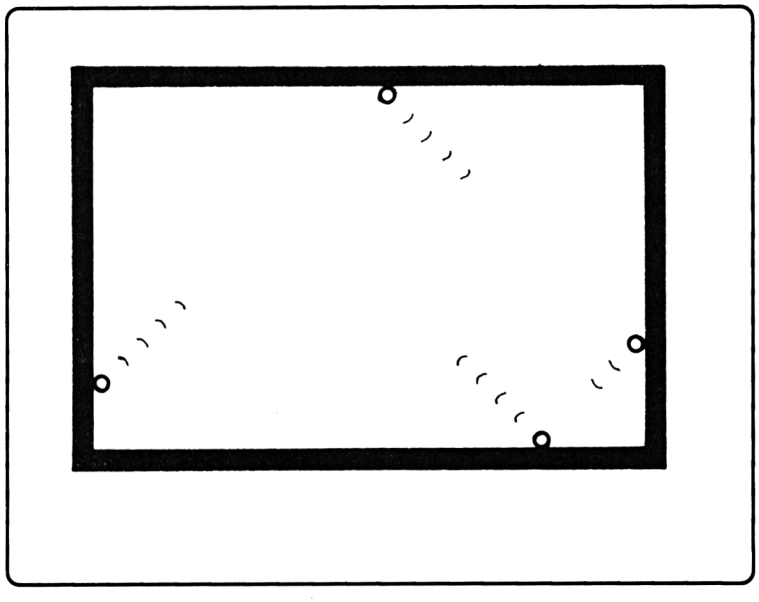
O. — Le pourtour du terrain est un peu long à écrire, mais ça y est, j'ai fini :



```
 10 FOR C=Ø TO 3Ø
 2Ø PRINT AT Ø,C;"■";AT 2Ø,C;"■
"
 3Ø NEXT C
 4Ø FOR L=Ø TO 2Ø
 5Ø PRINT AT L,Ø;"■";AT L,3Ø;"■
"
 6Ø NEXT L
 7Ø LET U=1
 8Ø LET V=1
 9Ø LET C=15
1ØØ LET L=1Ø
11Ø PRINT AT L,C;"  "
12Ø LET C=C+U
13Ø LET L=L+V
```

```
140 PRINT AT L,C;"O"
150 IF C=1 THEN LET U=-U
160 IF C=29 THEN LET U=-U
170 IF L=1 THEN LET V=-V
180 IF L=19 THEN LET V=-V
190 GOTO 110
```

P. — Tiens, tu as changé ta façon d'écrire les lignes de rebond !

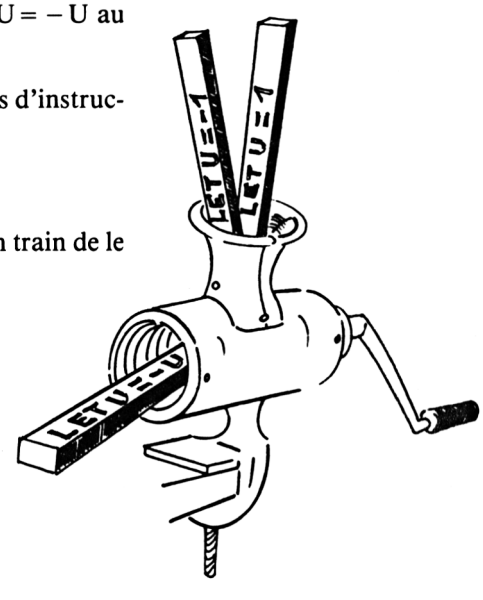


O. — Oui !... Je me suis dit que j'économisais ainsi de la mémoire, même si cela n'est plus indispensable maintenant avec la mémoire 16 K, en écrivant LET U = – U au lieu de LET U = – 1 ou LET U = 1 !

P. — Sais-tu que cette façon d'écrire va te permettre de gagner deux lignes d'instruction ?

O. — Ah bon ?... Comment cela ?

P. — En regroupant, deux par deux, les lignes 150 à 180, comme je suis en train de le faire :



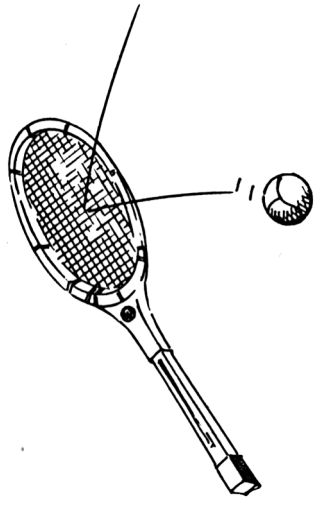

```
150 IF C=1 OR C=29 THEN LET U=-
U
170 IF L=1 OR L=19 THEN LET V=-
V
```

O. — Très astucieux !

P. — C'est toi qui m'en as donné l'idée, tu sais !

O. — Alors, en voici une autre !... Tu me montres comment s'effectue le rebond sur une raquette ?

Le rebond sur une raquette



P. — Je suppose que tu veux que l'angle du rebond dépende de la position de la raquette par rapport à la balle ?

O. — Bien sûr !

P. — Dans ces conditions, je te propose d'illustrer cela par un jeu de squash !... Cela te convient-il ?

O. — Tout à fait !... Je vais d'ailleurs te dessiner le terrain et la raquette !... Toi, tu me montreras comment on écrit le rebond !... D'accord ?

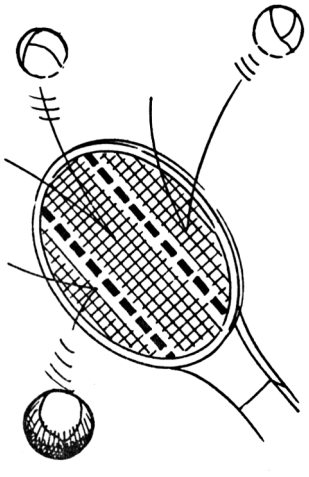
P. — D'accord !

O. — Alors, voici un terrain de squash :

```
10 FOR C=0 TO 16
20 PRINT AT 0,C;"■"
30 NEXT C
40 FOR L=0 TO 20
50 PRINT AT L,0;"■";AT L,16;"■
"
60 NEXT L
```

P. — Pas mal proportionné ton terrain !.... Maintenant, saurais-tu dessiner une raquette sur la ligne numéro 20 ?

O. — Sans problème !... Tu la veux de combien de carreaux de largeur ?



P. — Il me semble qu'il n'y a qu'une largeur de trois carreaux qui puisse convenir puisqu'il faut que le rebond de la balle soit différent selon la position de celle-ci par rapport à la raquette !

O. — Attends, je ne comprends pas bien !... Pourquoi trois carreaux ?

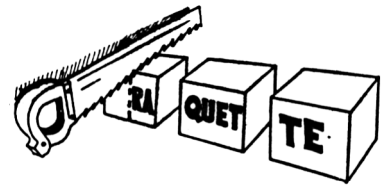
P. — Parce qu'il n'y a que trois rebonds possibles : tout droit lorsque la balle frappe le milieu de la raquette, en oblique vers la gauche lorsqu'elle frappe le côté gauche de celle-ci et en oblique vers la droite dans le cas contraire !

O. — Ah, c'est pour ça !... Alors, adjugé pour une raquette de trois carreaux :

```
70 LET C=6
80 LET L=20
90 PRINT AT L,C;"    "
100 IF INKEY$="5" AND C>1 THEN
LET C=C-1
110 IF INKEY$="8" AND C<13 THEN
LET C=C+1
120 PRINT AT L,C;"▬▬▬"
210 GOTO 90
```

P. — Tu ne trouves pas que la raquette est disproportionnée ?

O. — Si !... Deux carreaux de largeur seraient mieux, mais je ne vois pas comment on peut faire pour obtenir trois rebonds différents sur une raquette qui ne fait que deux carreaux de largeur ?



P. — Je crois que je viens de trouver la solution !... Il suffit de répartir les deux carreaux ainsi : un demi-carreau pour le rebond vers la gauche, un carreau entier pour le rebond tout droit et un demi-carreau pour le rebond vers la droite !

O. — Mais c'est bien sûr !... Ainsi, tout en conservant trois carreaux différents pour les rebonds, la raquette ne sera large que de deux !... J'écris cela tout de suite :

```
120 PRINT AT L,C;"▬ "
```

P. — Bien !... Reste maintenant à dessiner la balle !

O. — Ça, c'est facile !... La voici :

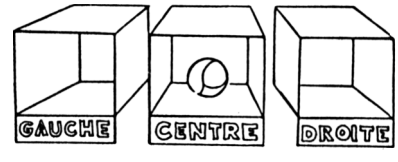
```
 82 LET U=1
 84 LET V=1
 86 LET X=7
 88 LET Y=3
130 PRINT AT Y,X;"  "
140 LET X=X+U
150 LET Y=Y+V
160 PRINT AT Y,X;"O"
```

P. — Bravo !... Mais, comme tu peux t'en rendre compte, il faut écrire les lignes d'instruction qui assurent le rebond sur les trois murs !

O. — Oui, bien sûr !... Les voici donc !... Tu noteras que j'en ai regroupées deux sous la même instruction :

```
170 IF Y=1 THEN LET V=-V
180 IF X=1 OR X=15 THEN LET U=-
U
```

P. — Parfait !... Il ne reste plus qu'à écrire le programme de rebond de la balle sur la raquette !



O. — Ça, ce n'est pas évident !... J'ai bien pensé à un « truc», mais je ne suis pas certain que cela marche !

P. — Dis toujours !

O. — Puisque la raquette fait trois carreaux de largeur, même si on n'en voit que deux, il suffit de détecter le moment où la balle arrive sur l'une des trois cases situées juste au-dessus de la raquette, et à commander alors le rebond !

P. — L'idée est bonne, quoique incomplète !... Commence donc à écrire le programme de rebond et je t'aiderai à le finir !

O. — Alors, le voici :

```
190 IF Y=19 AND X>=C AND X<=C+2
THEN LET V=-V
```

P. — Le début est bon, puisque tu n'autorises le rebond que si la balle arrive sur la raquette. Par contre, tu noteras que celui-ci est toujours le même, quelle que soit la zone d'impact de la balle sur la raquette !

O. — Eh oui, je sais !... C'est justement ça que je ne sais pas faire !

P. — Je vais te montrer !... D'abord, il faut faire appel à un sous-programme dès que tu détectes l'impact de la balle sur la raquette !... Cela oblige à réécrire la ligne 190 :

```
190 IF Y=19 AND X>=C AND X<=C+2
THEN GOSUB 220
```

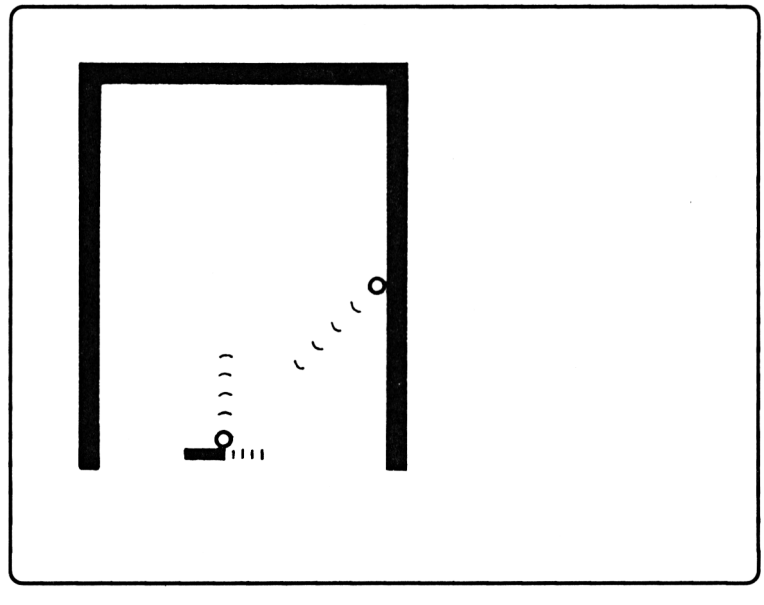
P. — ... ensuite, il faut écrire autant de lignes différentes que tu souhaites avoir de rebords différents !

O. — C'est à dire trois ?

P. — Exact !... Sans oublier, bien sûr, de placer en tête du sous-programme la ligne qui transforme V en $-V$, comme tu l'avais précédemment écrite !... Voici donc le sous-programme en question :

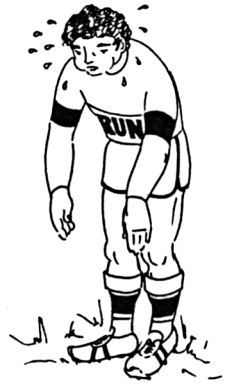
```
220 LET V=-V
230 IF X=C THEN LET U=-1
240 IF X=C+1 THEN LET U=0
250 IF X=C+2 THEN LET U=1
260 RETURN
```

O. — Super, ça marche !... Comme sur un vrai jeu !... Zut, je viens de « louper » la balle !... Dis donc, Papa, il n'y a pas moyen d'éviter de taper sur la touche RUN pour relancer le jeu ?... C'est fastidieux, et de plus, elle commence à fatiguer la pauvre !



P. — Si !... C'est même très simple !... Il suffit d'écrire une ligne du genre :

```
200 IF Y=21 THEN RUN
```



P. — ... sans oublier de placer en début de programme la fameuse instruction CLS, sinon les emplacements précédents de la balle et de la raquette ne seront pas effacés :

```
 5 CLS
```

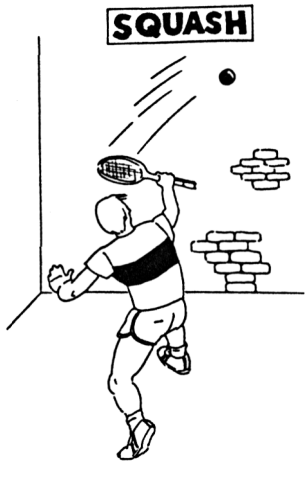
O. — Génial !... Si seulement on pouvait accélérer un peu la vitesse...

P. — S'il n'y a que cela pour te faire plaisir !... Remplace donc tes lignes 82, 84, 230 et 250 par les suivantes et dis-moi ce que tu en penses :

```
 82 LET U=2
 84 LET V=2
230 IF X=C THEN LET U=-2
250 IF X=C+2 THEN LET U=2
```

O. — Alors là, chapeau !... D'avoir augmenté la vitesse par deux, ça rend le jeu drôlement intéressant !

Et c'est ainsi que je vis Olivier se métamorphoser en champion de squash, luttant, la raquette à la main (ou presque), contre cette fichue balle qui n'arrêtait pas de rebondir contre les murs. Plus rien d'autre n'existait !... Du moins, c'est ce que je croyais jusqu'à ce qu'il m'annonce :



O. — Il y a comme un défaut !... Quand je coince la balle dans l'angle inférieur droit, entre la raquette et le mur, cette « couillonne » traverse le mur au lieu de rebondir dessus !... Ça vient de quoi, à ton avis ?

P. — Attends un instant que je réfléchisse !... Ah, je viens de trouver !... Cela provient du fait que la dernière instruction de rebond qu'elle reçoit, en provenance du sous-programme, est de partir vers la droite. Et c'est justement ce qu'elle fait !... Pour éviter cela, il faut placer la ligne d'appel du sous-programme avant les lignes de rebond sur les murs latéraux !

O. — Compris !... Quelque chose comme :

```
175 IF Y=19 AND X>=C AND C<=C+2
THEN GOSUB 220
190
```

P. — Exactement !... Et comme tu peux le constater, le défaut a disparu !

O. — A quoi ça tient quand même !

P. — Eh oui !... Ceci dit, j'ai terminé avec les différents rebonds. Si tu es prêt, on peut passer aux programmes d'impact !

O. — Tout de suite, si tu veux !

P. — D'accord, alors on va reprendre l'exemple qu'on avait laissé de côté, faute d'un programme d'impact, et on va le terminer ensemble !

VIDÉO INVERSÉE !

Eclatement d'un astéroïde

O. — Super idée !... Laisse-moi le temps de retrouver le programme de ce jeu et je suis à toi !...

P. — Tiens, je viens même d'avoir une autre idée !... Et si, plutôt que de recopier bêtement ce programme, on en profitait pour travailler en vidéo inversée ?

O. — Tu veux dire dessiner des fusées blanches sur un fond d'espace noir ?

P. — Oui !... Alors, qu'en penses-tu ?

O. — J'en pense que c'est une idée hyper géniale !...

P. — Veux-tu que je t'écrive le programme ?

O. — Non, non, laisse-moi faire !... Voici déjà pour le fond noir :

```
10 FOR L=0 TO 21
20 FOR C=0 TO 31
30 PRINT AT L,C;"■"
40 NEXT C
50 NEXT L
```

P. — Tu ne trouves pas que c'est un peu long ?

O. — Si !... Mais y-a-t-il moyen de faire autrement ?

P. — Oui !... Il existe une instruction qui permet d'accélérer par quatre la vitesse de fonctionnement de l'ordinateur !

O. — Ça, c'est intéressant !... Et elle s'intitule comment, cette instruction ?

P. — FAST !... Comme je viens de te l'expliquer, elle fait fonctionner l'ordinateur quatre fois plus vite, mais cela au détriment de l'affichage qui ne se fait plus !

O. — Alors, à quoi elle sert si on ne voit plus rien sur l'écran ?

P. — Elle sert, mais à condition d'être employée avec l'instruction SLOW !

O. — Je crois que je viens de comprendre !... L'instruction SLOW ramène l'ordinateur à sa vitesse normale !... En fin de compte, l'instruction FAST correspond à l'accélérateur et l'instruction SLOW au frein !

P. — C'est cela même !... J'ajouterai même qu'après une instruction SLOW, l'affichage réapparaît !

O. — Alors, je sais comment m'y prendre pour dessiner le fond noir plus rapidement ! Il suffit de faire précéder les lignes 10 à 50 d'une instruction FAST et de les faire suivre par une instruction SLOW :

```
  5 FAST
 55 SLOW
```

P. — Gagné !... Ainsi, le fond noir se dessine quatre fois plus vite !

O. — Je continue !... Voici donc les cinq fusées de l'escadre intergalactique :

```
 60 FOR A=6 TO 14 STEP 2
 70 PRINT AT A,0;"▄▀";AT A+1,0;
"▀"
 80 NEXT A
```

P. — Je les trouve un peu trop près du bord gauche de l'écran !

O. — Oui, tu as raison, ce n'est pas beau !... Je vais les décaler d'un carreau vers la droite :

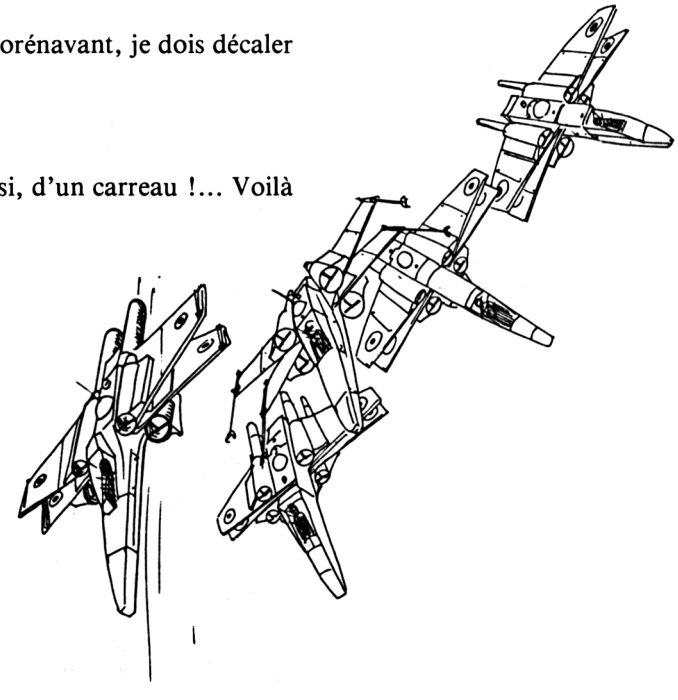
```
 70 PRINT AT A,1;"▄▀";AT A+1,1;
"▀"
```

P. — C'est bien mieux ainsi !

O. — Maintenant, il va falloir que je me souvienne que, dorénavant, je dois décaler le reste du programme d'un carreau vers la droite !

P. — Au tour de la fusée de tête !

O. — La voici !... J'en ai profité pour la décaler, elle aussi, d'un carreau !... Voilà ce que cela donne :



```
 90 LET L=10
100 PRINT AT L,5;"██";AT L+1,5;
"█"
110 IF INKEY$="6" AND L<14 THEN
LET L=L+1
120 IF INKEY$="7" AND L>6 THEN
LET L=L-1
130 PRINT AT L,5;"▄▀";AT L+1,5;
"▀"
```

P. — Sans oublier la ligne de fermeture de boucle !

O. — C'est vrai !... Décidément, je l'oublie toujours celle-là !... Je vais la numéroter 290 ; comme ça, j'aurais de la marge :

```
290 GOTO 100
```

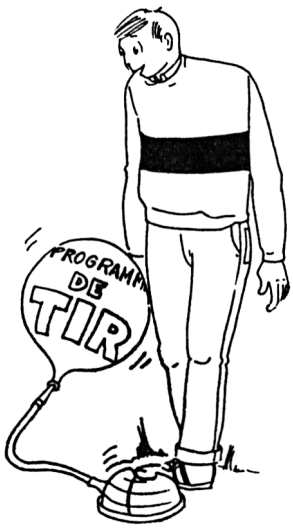
P. — Tout cela me paraît très bien !... La fusée de tête monte et descend correctement !... Adjugé !... On passe au programme de tir maintenant ?

O. — Oui !... J'en ai d'ailleurs profité pour le modifier légèrement par rapport au précédent, de façon à en simplifier l'écriture !

P. — Et sur quoi portent tes modifications ?

O. — Sur la position de la ligne qui augmente la valeur de X. Au lieu de la placer en dernier, après la ligne d'impression du point, je l'ai placée avant la ligne de commande du tir. Ainsi, il est possible d'écrire de façon similaire les lignes d'effacement et d'impression :

```
 92 LET X=6
 94 LET Y=10
140 IF X>6 THEN PRINT AT Y,X;"■
"
150 IF X>6 THEN LET X=X+1
160 IF X=32 THEN LET X=6
170 IF INKEY$="0" THEN LET X=7
180 IF X=7 THEN LET Y=L
190 IF X>6 THEN PRINT AT Y,X;"▪
"
```



P. — Très bien !... Il ne te reste plus qu'à terminer avec le programme d'apparition des astéroïdes et tu auras terminé ta part !

O. — Celui-là, il n'est pas bien compliqué !... Il suffit de le « pomper » bêtement :

```
 96 LET B=30
200 IF B=30 THEN LET A=INT (RND
*9)+6
210 PRINT AT A,B;"■"
220 IF B=0 THEN LET B=30
230 LET B=B-1
240 PRINT AT A,B;"✱"
```

P. — Pas si bêtement que tu veux bien le dire !... Je remarque que tu as permuté la position de deux lignes !

O. — C'est pour mieux effacer l'astéroïde quand il arrive sur le bord gauche de l'écran !

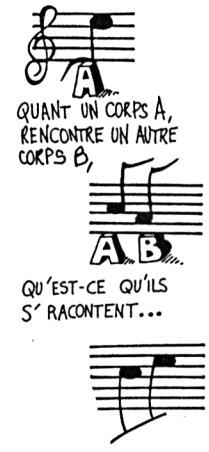
P. — Eh bien, maintenant, c'est à moi de t'écrire le programme d'impact sur les astéroïdes !... Je vais commencer par la ligne de détection de l'impact :

```
250 IF Y=A AND X=B THEN GOSUB 3
00
```

O. — Elle ressemble drôlement à celle du rebond sur la raquette !

P. — Et pour cause !... Dans les deux cas, il s'agit de détecter la rencontre d'un corps avec un autre !

O. — Pendant que tu me parlais, je me suis amusé à tirer sur les astéroïdes. Je ne voudrais pas te décevoir, mais ta ligne de détection des impacts n'en détecte qu'un sur deux en moyenne !



P. — Ah bon !... Tiens, d'où cela peut-il provenir ?... Ah, ça y est, j'ai trouvé !... C'est dû au fait que le projectile et l'astéroïde se déplacent tous les deux l'un vers l'autre et donc qu'ils peuvent se croiser sur la même case ou sur deux cases adjacentes !

O. — Ce qui fait qu'il faut effectuer la détection, non sur une seule case, mais sur deux !... Un instant, je te modifie la ligne 25Ø :

```
250 IF Y=A AND X=B OR Y=A AND X
=B+1 THEN GOSUB 300
```

P. — Pas mal !... Toutefois, il y a moyen d'écrire cela de manière plus simple !... Comme tu fais la détection sur deux cases adjacentes, il te suffit d'écrire les deux limites ainsi :

```
250 IF Y=A AND X>=B AND X<=B+1
THEN GOSUB 300
```

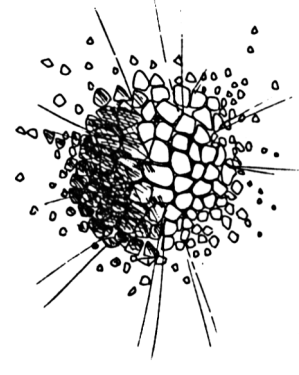
O. — Oui, tu as raison, c'est plus court !

P. — Surtout si tu dois faire une détection d'impact sur plus de deux cases adjacentes... comme c'est souvent le cas !

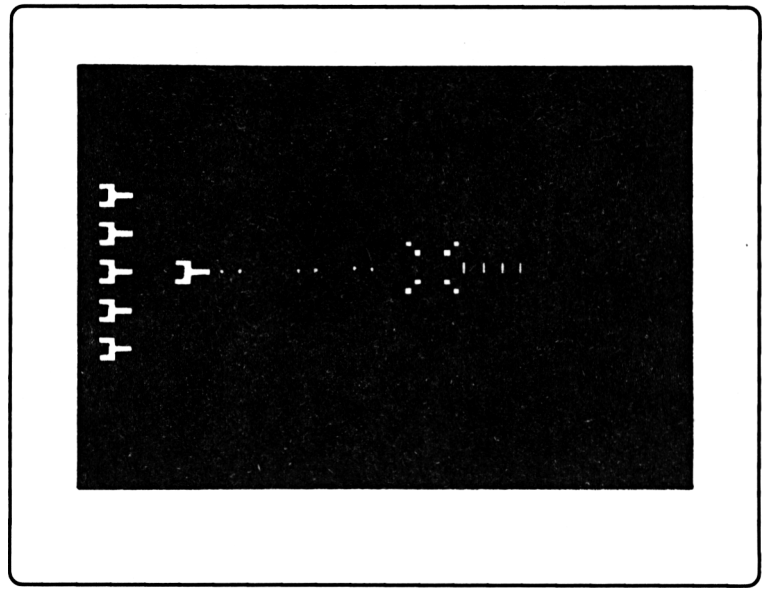
O. — Bon, ben alors, il ne reste plus qu'à écrire le programme d'impact !

P. — Je m'y emploie !... Tu noteras qu'en général, c'est long et fastidieux, mais tu noteras également que le résultat en vaut la peine !... Voici donc la chose :

```
300 PRINT AT A,B;"▒"
310 PRINT AT A,B;"■"
320 PRINT AT A-1,B-1;"▄▄▄";AT A
+1,B-1;"▀▀▀"
330 PRINT AT A-1,B-1;"▄▄▄";AT A
+1,B-1;"▀▀▀"
340 PRINT AT A-1,B-1;"▄▄▄";AT A
+1,B-1;"▀▀▀"
350 LET X=6
360 LET B=30
370 RETURN
```



O. — Génial !... D'accord, c'est long, mais c'est génial !... On dirait réellement que l'astéroïde éclate en mille morceaux qui se dispersent ensuite dans l'espace !... Par contre, une petite critique : l'astéroïde ne se déplace pas assez vite !



P. — Qu'à cela ne tienne !... Il suffit de modifier la ligne 23Ø ainsi :

```
230 LET B=B-2
```

P. — ... sans oublier, comme je viens de le faire, de modifier également les lignes 25Ø et 3ØØ :

```
250 IF Y=A AND X>=B AND X<=B+2
THEN GOSUB 300
300 PRINT AT A,B;"▒■"
```

O. — A quoi sert la modification de la ligne 3ØØ ?

P. — A effacer le projectile qui, dans un cas sur trois, demeure sur l'écran après l'impact !

O. — Ah bon !... Dis-donc, c'est drôlement mieux ainsi !... J'ai à peine le temps de me mettre en place pour tirer !... Loupé !... Je viens de louper l'astéroïde et celui-ci vient de percuter une des fusées de l'escadre !... Crois-tu qu'on puisse la faire se volatiliser elle aussi ?

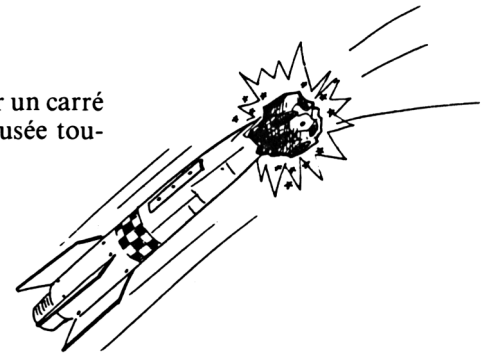
P. — Cela ne me paraît pas impossible !... Il suffit pour cela de détecter le moment où la variable B atteint cette fusée !

O. — Dans ce cas-là, c'est la valeur B = 2 qui convient !... Faut-il maintenant écrire un second sous-programme d'impact ?

P. — Non !... Tu peux te servir de celui existant, en incorporant cette valeur B = 2 dans la ligne 25Ø !

O. — Comme ceci ?

```
250 IF B=2 OR Y=A AND X>=B AND
X<=B+2 THEN GOSUB 300
```



P. — Tout à fait !... Par contre, je m'aperçois qu'il faut à nouveau rajouter un carré noir d'effacement à la ligne 300, de façon à faire disparaître la totalité de la fusée touchée :

```
300 PRINT AT A,B-1;"■▒■"
```

O. — Ça marche, sauf que, maintenant, le projectile n'est plus effacé après l'explosion d'une fusée !

P. — Décidément !... Je vois d'où cela provient !... Rajoute donc dans le sous-programme cette petite ligne d'effacement :

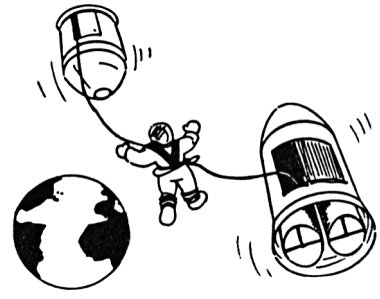
```
345 PRINT AT Y,X;"■"
```

O. — Ok, c'est bon !... C'est vraiment le plus chouette jeu que nous ayons réalisé depuis le début !

P. — Je suis content qu'il te plaise !... Quand tu auras fini d'y jouer, nous pourrons passer à un nouveau jeu, au moins aussi chouette, qui va nous permettre d'illustrer un autre type de sous-programme d'impact.

O. — Ah, ah !... Et c'est quoi ce nouveau jeu ?

P. — Un rendez-vous spatial !



Un rendez-vous spatial

O. — Ça peut-être intéressant !... Tu m'expliques un peu ?

P. — Bien sûr !... Imagine donc une capsule spatiale en orbite autour de la terre et sur laquelle doit s'arrimer une seconde capsule identique de façon à constituer un élément d'une station orbitale !

O. — J'imagine très bien la suite !... La seconde capsule étant pilotée, il faut arriver à la placer exactement à l'arrière de la première, en accélérant ou en ralentissant sa vitesse !

P. — Exact !... Maintenant, je ne sais pas si tu le sais, mais lorsqu'un corps tourne en orbite autour d'un autre, les lois de l'attraction universelle font qu'à une altitude donnée correspond une seule vitesse. Pour aller plus vite, il faut descendre, alors que pour aller moins vite, il faut monter !

O. — Dis donc, ça ne facilite pas l'arrimage tout ça !

P. — C'est justement parce que ce n'est pas évident, que ce jeu présente de l'intérêt !... De surcroît, si l'arrimage est loupé, les deux capsules se heurtent violemment et retombent sur terre !

O. — Effectivement, la manœuvre est délicate !

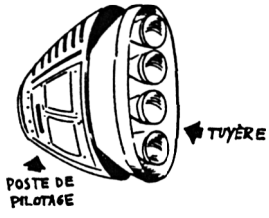
P. — Le programme aussi !...

O. — Bon !... Alors par quoi je commence ?

P. — Par écrire le programme d'une première capsule qui navigue dans l'espace !

O. — Ça, c'est facile !... La voici :

```
 10 LET L=10
 20 LET C=0
 50 PRINT AT L,C;"      "
 60 LET C=C+1
 70 IF C=29 THEN LET C=0
 80 PRINT AT L,C;">▀▀O"
190 GOTO 50
```



P. — Pas mal ta capsule spatiale, avec sa tuyère et son poste de pilotage !

O. — Ouais !... J'avoue être assez content de moi !

P. — Je constate également que tu as prévu large dans la numérotation de tes lignes !

O. — C'est exprès !... Comme tu m'as dis qu'il fallait dessiner deux capsules différentes, j'ai prévu le coup !

P. — Bien !... Alors, puisque tout est prévu, je te laisse continuer d'écrire la suite du programme pour la seconde capsule !... Je te rappelle qu'il faut qu'elle puisse monter et descendre !

O. — De combien de lignes ?

P. — Je pense que 5 lignes au-dessus et 5 lignes en dessous devraient suffire !

O. — Ok !... Alors j'y vais !... Je suis pratiquement en train de réécrire le même programme en changeant les coordonnées L, C en Y et X et en rajoutant deux lignes de changement d'altitude :

```
 30 LET Y=5
 40 LET X=0
 90 PRINT AT Y,X;"      "
100 IF INKEY$="6" AND Y<15 THEN
LET Y=Y+1
110 IF INKEY$="7" AND Y>5 THEN
LET Y=Y-1
120 LET X=X+1
130 IF X=29 THEN LET X=0
140 PRINT AT Y,X;">▀▀O"
```

P. — Eh bien bravo !... Il ne te reste plus qu'à faire varier la vitesse de cette capsule en fonction de son altitude !

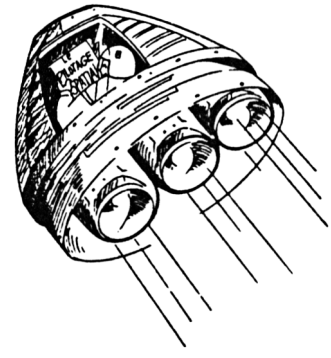
O. — Je vais t'avouer que je ne vois pas très bien comment !... Ce qui est certain, c'est que les deux vitesses doivent être identiques lorsque cette seconde capsule est dans l'alignement de la première !

P. — Exact !... Et sur quelle ligne cela se produit-il ?

O. — Uniquement sur la ligne 10 !

P. — Alors, voici la solution !... Il suffit de remplacer ta ligne d'instruction 120 par celle-ci :

```
120 LET X=X+Y/10
```



O. — Très astucieux tout ça !... Ainsi, lorsque la capsule monte, sa vitesse diminue jusqu'à la moitié de celle de l'autre lorsqu'elle est au plus haut, alors qu'elle augmente en descendant, jusqu'à valoir une fois et demie celle de l'autre lorsqu'elle est au plus bas !

Et Olivier se découvrit une passion pour le pilotage d'une capsule spatiale, la faisant changer d'orbite constamment de façon à accélérer ou à ralentir sa vitesse. Je le ramenais sur terre en lui déclarant :

P. — Tu ne remarques rien de gênant ?

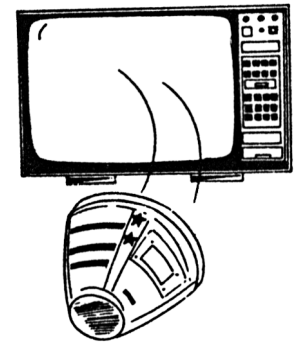
O. — Si, justement je voulais t'en parler !... Les deux capsules n'avancent pas en même temps !... Ça ne facilite pas l'arrimage tout ça !

P. — Ce petit défaut est uniquement dû au fait que tu imprimes la première capsule à la ligne 80 et la seconde à la ligne 140 !... Comme ces deux lignes ne sont pas placées l'une à la suite de l'autre, il s'écoule fatalement un certain temps entre les deux !

O. — S'il n'y a que ça, je vais vite l'arranger ! Et voilà le boulot !... Qu'est-ce que tu en penses ?

```
 50 PRINT AT L,C;"      "
 60 PRINT AT Y,X;"      "
 70 IF INKEY$="6" AND Y<16 THEN
LET Y=Y+1
 80 IF INKEY$="7" AND Y>5 THEN
LET Y=Y-1
 90 LET C=C+1
100 LET X=X+Y/10
110 IF C=29 THEN LET C=0
120 IF X=29 THEN LET X=0
130 PRINT AT L,C;">■■O"
140 PRINT AT Y,X;">■■O"
```

P. — J'en pense que tu te débrouilles drôlement bien maintenant !



O. — Tiens, mon jeu n'est pas encore tout à fait au point !... J'obtiens un message d'erreur B/140 !

P. — Je sais d'où cela provient !... C'est tout bonnement parce que tu demandes à l'ordinateur d'imprimer ta seconde capsule en dehors de l'écran !

O. — Et pourquoi la ligne de détection de la sortie d'écran ne joue pas son rôle ?

P. — Parce que tu ne progresses pas par valeurs entières de carreau lorsque tu es à une altitude différente de celle de la première capsule !

O. — Compris !... Il faut donc que je modifie ainsi la ligne de détection de la sortie d'écran :

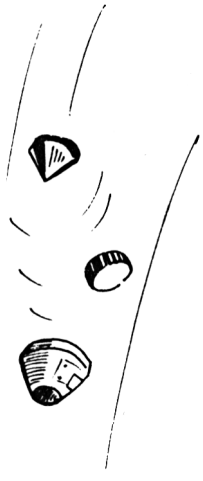
```
120 IF X>=29 THEN LET X=0
```

P. — C'est presque correct !... Arrime donc les deux capsules et dis-moi ce que tu constates !

O. — Un instant !... C'est pas si facile que ça à faire !... Ça y est, j'ai réussi !... Zut, après la réapparition des deux capsules sur la gauche de l'écran, il y a maintenant un espace d'un carreau entre elles !... C'est ça que tu voulais que je constate ?

P. — Oui !... As-tu une idée d'où cela peut provenir ?

O. — Non, aucune !



P. — Cela provient encore une fois de la ligne 120 !... Je ne sais si tu t'en souviens, mais lorsqu'une des coordonnées d'affichage n'est pas entière, l'affichage sur l'écran se produit dans le carreau correspondant ou dans le suivant, selon que la partie décimale de la variable est inférieure ou supérieure à .05 !

O. — Effectivement, cela me rappelle quelques vagues souvenirs !... Donc, si j'ai bien compris, il faut détecter la sortie d'écran, non plus à X > = 29, mais à X > = 28.5 !... Comme cela, il n'y a plus de risque de perdre un carreau à la remise à zéro des variables !

```
120 IF X>=28.5 THEN LET X=0
```

P. — Bravo !... Maintenant que tu as terminé le jeu, il ne nous reste plus qu'à écrire le programme d'impact et de chute libre !

O. — Là, j'avoue que je préfère te laisser faire !... Déjà, rien que la détection de l'impact...

P. — Tu vas voir, ce n'est pas sorcier !... On va progresser étape par étape, de façon que tu comprennes bien !

O. — Ok !

P. — Lors de l'arrimage de la seconde capsule à l'arrière de la première, la coordonnée horizontale X de cette seconde capsule est égale à celle de la première, c'est-à-dire à C, diminuée de 4 carreaux !... Donc, tant que X reste inférieur ou égal à C – 4, on considérera qu'il n'y a pas d'impact entre les deux. On peut donc écrire la ligne de détection de l'impact ainsi :

```
150 IF Y=L AND X>=C-4 THEN GOTO
200
```

O. — Très bien !... C'est donc tout simple !

P. — Pas tant que cela !... Arrime-toi et dis-moi ce qui se passe !

O. — Voici, ça y est !... Tiens, le programme se bloque, comme s'il y avait eu impact !... Ce n'est pourtant pas le cas !... Je suis bien à quatre carreaux derrière la première capsule !

P. — Nous retrouvons ici exactement le même problème que précédemment, à savoir

qu'il suffit que la valeur de X soit décimale pour que l'écart entre X et C soit plus petit que 4, ce qui est assimilé à un impact. Il faut réécrire la ligne ainsi pour ne plus avoir ce problème :

```
150 IF Y=L AND X>=C-3.5 THEN GO
TO 200
```

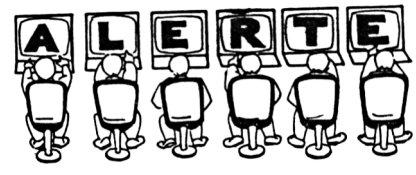
O. — Effectivement, ça marche !... On peut écrire le programme de chute libre ?

P. — Non, pas encore !... Amuse-toi à faire le tour de la première capsule en passant en dessous puis en repassant au-dessus !

O. — T'as raison, il y a comme un défaut !... Quand la seconde capsule passe devant la première, le programme se bloque à nouveau !

P. — C'est tout simplement parce que la condition de détection d'un impact est remplie, à savoir que X est plus grand que C. Il faut donc limiter cette condition vers l'avant et cela peut être fait ainsi :

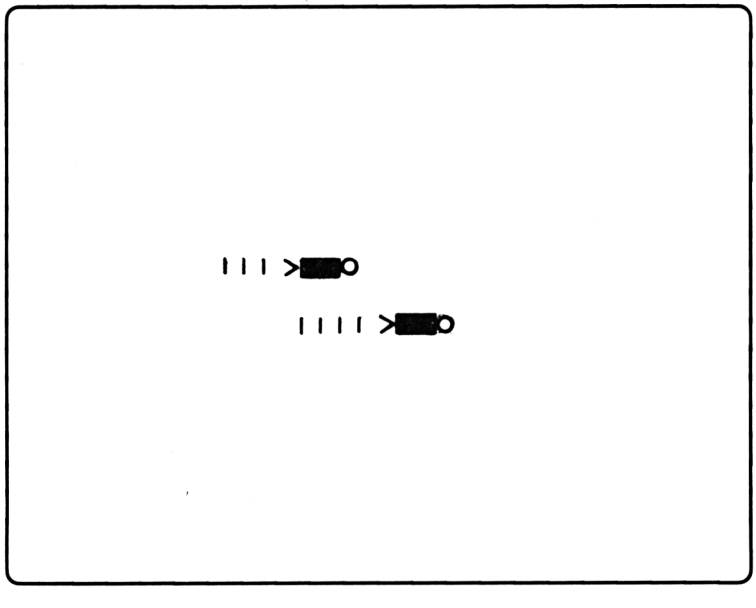
```
150 IF Y=L AND X>=C-3.5 AND X<=
C+3.5 THEN GOTO 200
```



O. — C'est bon !... Ça permet même l'arrimage à l'avant de la première capsule !... Enfin, c'est fait !... On va pouvoir passer au programme de chute libre !

P. — Quelle impatience !... Le voilà ton programme :

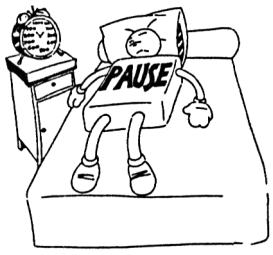
```
200 PRINT AT L,C;"      "
210 PRINT AT Y,X;"      "
220 LET L=L+1
230 LET Y=Y+1
240 IF L=22 THEN GOTO 280
250 PRINT AT L,C;">▀▀O"
260 PRINT AT Y,C;">▀▀O"
270 GOTO 200
```



O. — Super génial !... Si je loupe l'arrimage, c'est la chute au sol qui m'attend !... Heureusement qu'il y a des parachutes...

P. — Quand tu pourras m'accorder quelques secondes, tu réfléchiras à la manière de terminer ce jeu !... J'ai volontairement écrit une ligne de sortie de boucle pour cela !

O. — Je ne sais pas trop !... On pourrait peut-être redémarrer le jeu après un petit temps de pause, histoire de récupérer un peu !



P. — Pourquoi pas ?... Il faut pour cela faire appel à l'instruction PAUSE qui, comme son nom l'indique, sert à « perdre du temps ». Sa syntaxe d'écriture veut qu'on la fasse suivre d'un nombre qui indique quelle doit être la durée de la pause !

O. — Et à quoi correspond ce nombre ?... A des secondes ?

P. — Non !... A des 1/50e de seconde !

O. — Quelle valeur bizarre !

P. — Pas tant que cela, quand tu sais que la fréquence du secteur alternatif est de 50 hertz !

O. — Ah, ça vient de là !... Alors, si j'ai bien compris, pour une pause de 2 secondes, il faut écrire :

```
280 PAUSE 100
```

P. — Exactement !... Maintenant, il faut que je te signale que l'emploi d'une telle instruction présente ce qui, parfois, peut-être considéré comme un inconvénient !

O. — Ah !... Et c'est quoi ?

P. — Elle fait décrocher l'image du téléviseur !... Cela n'est pas gênant en fin de jeu, mais cela pourrait l'être si tu décidais de l'employer dans une boucle de programme pour ralentir son déroulement par exemple !

O. — A n'utiliser qu'avec parcimonie, si je comprends bien !

P. — Eh oui !... Maintenant, pour en revenir au jeu et le terminer, je te propose d'écrire tout simplement :

```
290 RUN
```

O. — Et voilà un autre super jeu qui marche !... Quel va être le prochain ?

P. — Décidément, tu es insatiable !... Le prochain ne sera pas un jeu, car il faut que je te parle en détail de l'instruction PEEK !

L'instruction PEEK

O. — L'instruction quoi ?

P. — PEEK !... Cette instruction permet de connaître le contenu d'une case-mémoire. Il suffit pour cela de rajouter le numéro de la case mémoire à sa suite !

O. — C'est bien beau tout ça, mais à quoi est-ce que ça peut servir ?

P. — A détecter la présence d'un symbole particulier sur l'écran, comme un astéroïde dans le cas d'un impact, un mur dans le cas d'un rebond, un obstacle dans le cas d'une course de voitures, etc...

O. — Mais elle est très intéressante, cette instruction PEEK !

P. — Je te crois !... Seulement, son emploi n'est pas évident !... Il faut bien la maîtriser et c'est pour cela que nous allons l'étudier bien en détail !

O. — Puisque c'est toi qui le dis !... Par quoi on commence ?

P. — Par écrire ceci !... Je t'expliquerai ensuite à quoi cela va nous servir !

```
10 PRINT PEEK 16396
20 PRINT PEEK 16397
```

O. — Il y a deux nombres qui s'inscrivent en haut de l'écran : 161 et 64 ?... A quoi correspondent-ils ?

P. — Au nombre de cases-mémoire utilisées dans la mémoire de programmation. Il faut que je t'explique que la mémoire de programmation, qu'elle soit de 16K ou de 1K, est divisée en pages de 256 cases-mémoires. Ainsi, le premier nombre affiché (161) correspond au nombre de cases-mémoire utilisées dans la page 64. Quant aux nombres particuliers 16396 et 16397, il s'agit de variables dites « variables système » dont la valeur correspond justement au nombre de cases-mémoire employées pour la première et la page concernée pour la seconde !

O. — Je commence à comprendre !... Ainsi, au fur et à mesure que je tape mon programme, je peux vérifier la quantité de mémoire qu'il occupe !

P. — Exactement !... Je dirais même plus, cela te permet de vérifier si telle façon d'écrire est plus économique qu'une autre, information bien utile quand tu ne possèdes que la petite mémoire 1K d'origine !

O. — Mais cela me donne une idée !... Je vais profiter de cette instruction PEEK pour déterminer combien de cases-mémoire emploie chaque instruction !

P. — Bonne idée !... Commence donc par :

```
30 LET A=U
```

O. — Le premier nombre est passé de 161 à 170 tandis que le second est toujours de 64 !... C'est normal ça ?

P. — Tout à fait !... Cela veut dire que ce que tu viens d'écrire a consommé 9 cases-mémoire de plus !

O. — C'est donc aussi simple que ça ?... Mais dis donc, 9 cases-mémoire pour si peu de chose, c'est pas un peu beaucoup ?

P. — Pas tellement si tu analyses en détail cette ligne d'instruction. En fait, le numéro de la ligne et l'instruction LET consomment 6 cases-mémoire à eux deux !... Pour le reste, chaque lettre et chaque signe de ponctuation consomment une case-mémoire chacun !

O. — Est-ce que c'est pareil pour les autres instructions ?

P. — Oui !... Quelles que soient l'instruction et le nombre de lettres qu'elle comporte, elle emploie invariablement 6 cases-mémoire !

O. — Et les chiffres !... Je vais pouvoir vérifier si c'est vrai qu'un chiffre occupe plus de mémoire qu'une lettre !

P. — Excellente occasion !... Ecris donc à la place de la ligne précédente :

```
30 LET A=1
```

O. — Hou là là !... Le nombre de cases employées est passé de 170 à 176 !

P. — Ce qui, concrètement, revient à dire qu'un seul misérable petit chiffre occupe à lui tout seul 7 cases-mémoire !

O. — Tu parles d'un gaspillage !... Et si j'écris un nombre de deux chiffres, tu crois que cela va occuper 14 cases-mémoire ?

P. — Essaye donc pour voir !

O. — C'est vrai !... Le temps de poser la question :

```
30 LET A=12
```

O. — ... et la réponse est 8 !... Un nombre de deux chiffres occupe 8 cases-mémoire, c'est-à-dire une de plus qu'un chiffre tout seul !

P. — Et c'est normal, puisque c'est justement pour caser l'éventuelle partie décimale de chaque nombre que tant de cases-mémoire ont été réservées !

O. — Je comprends maintenant pourquoi il valait mieux employer des lettres que des chiffres quand on cherchait à économiser la mémoire !

P. — N'est-ce-pas ?... Bien !... Ce point étant éclairci, il nous reste à présent à voir en détail comment il est possible d'analyser le contenu d'un point précis de l'écran, grâce à cette instruction PEEK !

O. — Tu peux m'expliquer, car je ne saisis pas bien ce que tu veux dire !

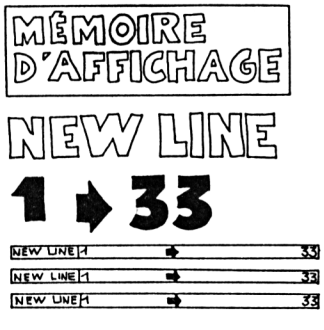
P. — Bien sûr !... Imagine un jeu dans lequel tu dessines un symbole, une étoile par exemple, dans une des 704 cases possibles de l'écran, et que ton adversaire doive trouver dans quelle case tu l'as mise !

O. — Un peu comme à la bataille navale ?

P. — Si tu veux !

O. — Ça ne va pas être facile à faire, cette recherche !... Comment est-ce qu'on fait ?

P. — Pour que tu en comprennes le principe, il faut que je t'apprenne que la mémoire d'affichage débute à la suite de la mémoire de programmation !... Ainsi, si tu en es rendu à la case-mémoire 171 de la page 64 pour ton programme, la case suivante, c'est-à-dire la 172 de la même page, sera la première case de la mémoire d'affichage !



O. — Je commence à voir émerger une faible lueur de compréhension !

P. — Bon, alors je continue !... Si maintenant je te dis que la mémoire d'affichage est structurée par lignes et que chaque ligne comporte 33 cases-mémoire, la première étant réservée à une instruction NEW LINE, qu'est-ce que tu me réponds ?

O. — Je te réponds qu'il me suffit de connaître l'emplacement qu'occupe dans la mémoire 16K la case-mémoire qui correspond au carreau de coordonnées (Ø, Ø), et je te trouve ce que tu veux où tu veux !

P. — En d'autres termes, il faut que tu connaisses l'adresse de cette case-mémoire !... Or, cette adresse s'obtient facilement quand tu connais celle de la dernière case-mémoire employée par la mémoire de programmation : c'est celle qui la suit !

O. — Et comment fait-on pour déterminer l'adresse de cette dernière case-mémoire ?

P. — En multipliant le nombre de pages par 256, puisqu'il y a 256 cases-mémoire par page et en y ajoutant le nombre de cases-mémoire employées sur la dernière page en cours d'utilisation !

O. — Attends !... Si j'ai bien compris, et en prenant l'exemple précédent (171, 64), j'obtiens pour l'adresse de cette dernière case : 256 × 64 + 171 = 16555 !

P. — C'est tout à fait exact !

O. — Donc l'adresse suivante, c'est-à-dire la 16556, est celle de l'instruction NEW LINE de la première ligne de la mémoire d'affichage !

P. — Encore exact !...

O. — Autrement dit, pour obtenir l'adresse de cette première case-mémoire, je dois écrire : 1 + 256*PEEK 16397 + PEEK 16396.

P. — Bravo !... Et si tu veux accéder à la case-mémoire qui correspond au point de coordonnées (10, 15), qu'est-ce que tu écris ?

O. — Je rajoute à l'expression précédente la quantité de cases-mémoire qui séparent le point d'origine (0,0) de celui qui nous intéresse !... Ça donne ceci : 10*33 + 15 + 1 + 256*PEEK 16397 + PEEK 16396.

P. — Je crois qu'il est temps qu'on vérifie tout cela avec un petit exercice pratique !... Tape donc ceci :

```
10 PRINT AT 10,15;"*"
```

P. — ... suivi du petit programme de recherche suivant :

```
20 FOR L=0 TO 21
30 FOR C=0 TO 31
40 LET A=1+C+33*L+256*PEEK 163
97+PEEK 16396
50 IF PEEK A=23 THEN PRINT L;"
,";C
60 NEXT C
70 NEXT L
```

O. — ... petit programme que j'essaye de comprendre !... Tout d'abord, les lignes 20, 30, 60 et 70 permettent au programme d'explorer tous les points de l'écran !... La ligne 40 correspond à la recherche de l'adresse de chaque case-mémoire explorée !... Reste la ligne 50 que je ne comprends pas !

P. — C'est le contraire qui m'aurait étonné !... Lorsque tu utilises l'instruction PEEK pour connaître le contenu d'une case-mémoire, tu obtiens en réponse un nombre compris entre 0 et 255. Or la signification de ce nombre n'est pas la même selon l'adresse employée. Tu as vu que pour les adresses 16396 et 16397 qui étaient celles des variables système, leurs valeurs correspondaient au nombre de cases-mémoire employées sur la dernière page en cours de remplissage. Pour les adresses de la mémoire d'affichage, ce nombre correspond au code du caractère employé.

O. — J'imagine donc que le nombre 23 correspond au code du signe * !

P. — N'imagine pas !... Vérifie-le à la page 181 de ton manuel d'utilisation !

O. — C'est ça !... Le code 23 correspond au signe * !

P. — Alors, puique tu as tout compris, lance ton programme de recherche pour voir s'il retrouve les coordonnées exactes de l'étoile !

O. — C'est fait !... Qu'est-ce que c'est long !... Ça y est, ça marche !... J'obtiens 1Ø, 15 !

P. — Amuse-toi à changer de coordonnées et de signe pour vérifier si la méthode est valable dans tous les cas !

O. — Je m'y emploie !... Je viens d'ailleurs d'avoir une idée concernant l'écriture de la ligne 5Ø !... Si mes souvenirs sont exacts, il n'est pas nécessaire que je fouille dans mon manuel pour trouver le code du caractère que j'ai employé. Il suffit que j'écrive :

```
5Ø IF PEEK A=CODE "*" THEN PRI
NT AT L;",";C
```

P. — Bravo !... Ta solution est très astucieuse !... Elle évite effectivement de passer par le code des caractères !

O. — Bon !... Eh bien si on a fini avec cette instruction PEEK, on pourrait peut-être passer à un jeu qui l'emploie !

P. — Bonne idée !... Il me reste toutefois à préciser un petit point de détail !

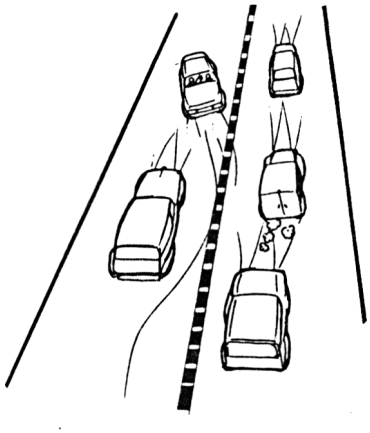
O. — Fais vite, qu'on puisse passer à un nouveau jeu !

P. — Il s'agit de l'écriture de la ligne 4Ø !... Elle peut être différente lorsque tu emploies seulement une mémoire 1K. Dans ce cas, et pour des raisons évidentes d'économie, seules les lignes employées sont à prendre en compte. Si, par exemple, ton jeu démarre sur la cinquième ligne, il n'y a pas lieu de tenir compte des quatre lignes précédentes. De même, si tu n'emploies qu'un certain nombre de colonnes, il n'y a pas lieu de tenir compte des colonnes situées au-delà de la dernière colonne employée. Ainsi, si tu limites ton jeu aux vingt premières colonnes de l'écran, il ne faut pas prendre en compte les douze suivantes de chaque ligne !

O. — C'est donc pour ça qu'on évitait la saturation de la mémoire d'affichage en travaillant dans la partie gauche de l'écran !

P. — Eh oui !... Maintenant que tout ceci est bien au point, je te propose de mettre en pratique ce que tu viens d'apprendre dans une course de voitures !

O. — Au « poil » !



Obstacles sur une route

P. — Je te propose donc de dessiner une voiture qui roule sur une route et qui doit slalomer entre des voitures qui arrivent en sens inverse !... Qu'en penses-tu ?

O. — Ça me paraît très chouette !... Et que se passe-t-il quand deux voitures se tamponnent ?

P. — C'est justement cet aspect du jeu qui est nouveau et qu'il va falloir développer ensemble !... Un petit dérapage qui amènerait les deux voitures en travers de la route, cela te conviendrait-il ?

O. — Tu penses si ça me convient !... Ça va être super !

P. — Bon, alors au boulot !... La première des choses à dessiner, c'est la route !... Tu t'en occupes ?

O. — Oui !... Je me souviens encore comment il faut faire !... Tu as une préférence pour les lignes blanches ?

P. — Oui justement !... Pour diverses raisons que tu saisiras plus tard, je voudrais une ligne blanche interrompue, chaque portion blanche faisant trois carreaux de longueur et l'intervalle entre deux étant de deux carreaux !

O. — Compris !... Et une route qui marche, une !

```
10 FOR A=1 TO 3
20 PRINT AT 21,0;"▌ ▌▌▌
```

être blanches sur fond noir !... Il faudra que je me souvienne de ça lorsque j'écrirai le programme de la voiture qui va slalomer entre toutes ces voitures obstacles !

P. — Tu ne vas pas avoir longtemps à attendre pour cela, car nous y sommes !

O. — Ah !... Ce coup-ci, ce qui me gêne, ce n'est pas d'écrire le programme de déplacement d'une voiture, ça je sais le faire, c'est plutôt de trouver l'endroit où il faut le placer !

P. — Effectivement, ce n'est pas simple !... Comme il faut pouvoir déplacer la voiture après chaque instruction SCROLL et qu'il y a deux instructions SCROLL, je ne vois que la solution qui consiste à faire appel à un sous-programme !

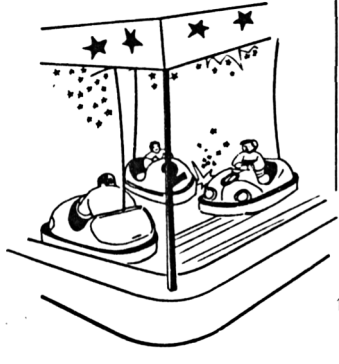
O. — C'est vrai qu'un sous-programme arrange tout !... Il suffit que j'écrive après chaque instruction SCROLL :

```
35 GOSUB 100
75 GOSUB 100
```

P. — Comme cela, tu accèdes au sous-programme à chaque parcours de chacune des boucles !

O. — Maintenant, au sous-programme de déplacement latéral de la voiture ! Ça, ce n'est pas compliqué :

```
100 PRINT AT 9,C;"■■";AT 10,C;"
■■"
110 IF INKEY$="5" AND C>2 THEN
LET C=C-1
120 IF INKEY$="8" AND C<8 THEN
LET C=C+1
130 PRINT AT 10,C;"■ ■";AT 11,C;
"■ ■"
```



O. — Tu remarqueras que j'ai pensé à effacer une ligne plus haut à cause de l'instruction SCROLL !

P. — J'ai remarqué !... J'ai aussi remarqué que tu avais oublié la ligne RETURN et la ligne de définition de la variable C !

O. — Hum !... Si nous parlions d'autre chose !... Les voici les deux lignes qui manquent :

```
140 RETURN
  5 LET C=5
```

P. — Bien !... Il ne nous reste plus que le plus délicat à écrire, à savoir la ligne de détection de l'impact entre voitures !

O. — Ça, je te laisse le faire !... J'essayerai tout seul la prochaine fois !

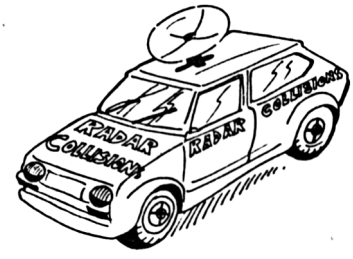
P. — Ok !... Commençons donc par déterminer la ligne de l'écran sur laquelle nous allons effectuer la détection de l'impact !

O. — C'est obligatoirement la ligne 11, puisqu'elle correspond à l'avant de la voiture que je conduis !

P. — C'est cela !... Ensuite, il faut écrire la ligne par laquelle tu vas détecter le contenu des deux cases-mémoire qui correspondent à l'avant de la voiture !

O. — Ben !... C'est idiot comme idée !... Ça va fatalement être le dessin de la voiture, avec son moteur et ses roues !

P. — Ta remarque est très judicieuse !... Cela pourrait être ce que tu dis si nous effectuions la recherche après avoir imprimé l'avant de la voiture sur l'écran !... Par contre, si c'est l'inverse, nous ne trouvons que la route ou l'avant d'une voiture obstacle !



O. — Vu !... Il faut donc séparer en deux la ligne d'instruction 130 et insérer la ligne de détection entre les deux nouvelles lignes ainsi créées !

P. — Quant à la ligne de détection, il n'y a qu'à reprendre ce que nous avons étudié précédemment, c'est-à-dire :

```
 130 PRINT AT 10,C;"■ ■"
 132 LET B=1+C+33*11+256*PEEK 16
397+PEEK 16396
 139 PRINT AT 11,C;"■ ■"
```

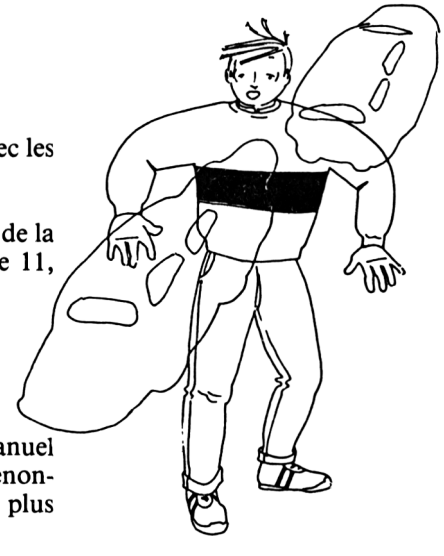
O. — Ah bon !... Ce n'est pas plus compliqué que cela ?

P. — Normalement non !... Il n'y a plus qu'à reprendre la valeur de B ainsi définie et à l'employer pour détecter l'impact de face, sur l'aile gauche ou sur l'aile droite.

O. — Et tu fais ça comment ?

P. — Comme ceci !... Voici déjà pour l'impact de face :

```
 134 IF PEEK B=CODE "■" THEN GOT
O 200
```



O. — Mais ça ne marche pas ton « truc » !... J'ai beau me présenter de face avec les voitures obstacles, je leur passe à travers comme si elles étaient des fantômes !

P. — Tiens, me serais-je trompé ?... Pourtant, il y a bien onze lignes au-dessus de la ligne 11 !... Et donc la case-mémoire qui correspond à l'avant de la voiture, ligne 11, colonne C, a bien pour adresse :

1 + C + 33 * 11 + 256 * PEEK 16397 + PEEK 16396. Là, je ne comprends plus !

O. — Moi non plus !

P. — Un instant, je crois que je viens d'avoir une idée !... Passe-moi donc le manuel d'utilisation !... Il me semble avoir lu quelque part que la règle générale que je t'ai énoncée, à savoir que la mémoire d'affichage est constituée de lignes de 33 cases, n'est plus vraie quand on emploie l'instruction SCROLL !

O. — Et alors ?

P. — Mon idée était bonne !... Effectivement, à la page 172, il est dit que dans le cas d'emploi de l'instruction SCROLL, la règle de structuration de la mémoire d'affichage est celle de la mémoire 1K !

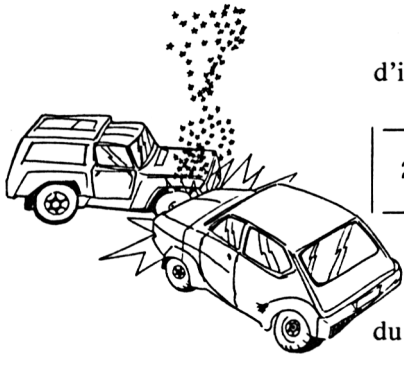
O. — Ce qui voudrait dire que les onze lignes ne comportent que 13 cases-mémoire chacune !

P. — Normalement, oui !... Tu essayes pour voir ?

O. — Tout de suite !... Le temps de modifier la ligne 132 :

```
132 LET B=1+C+13*11+256*PEEK 16
397+PEEK 16396
```

O. — ... et ça marche !... Ouf !... Au sous-programme d'impact, maintenant !... Tu me l'écris ?



P. — Bien sûr !... On va d'abord commencer par dessiner des étoiles aux points d'impact, histoire de bien montrer qu'il y a eu impact :

```
200 PRINT AT 11,C;"**"
```

P. — ... puis on va faire reculer chaque voiture d'une case, de façon à simuler l'effet du choc :

```
210 PRINT AT 9,C;"■ ■";AT 13,C;"
■ ■"
```

... et pour terminer, on va effacer les étoiles :

```
220 PRINT AT 11,C;"  "
```

O. — C'est pas trop mal, mais c'est un peu trop rapide à mon goût !... On ne voit pas très bien ce qui se passe !

P. — Qu'à cela ne tienne !... Il suffit de rajouter une petite pause après la ligne 200 et cela devrait te plaire !

O. — Ça, je sais le faire !... Avec une demi-seconde, cela devrait suffire :

```
205 PAUSE 25
```

P. — T'avais raison !... C'est nettement mieux ainsi !

O. — Pendant que tu écrivais ce programme, j'ai pensé à quelque chose !

P. — Je t'écoute !

O. — Lors de l'impact sur le côté droit ou gauche de la voiture, on pourrait essayer de dessiner une sorte de dérapage contrôlé !... Ça doit être faisable !

P. — Tu peux même dire que c'est une très bonne idée !... Si tu veux l'écrire tout seul, je te conseille de l'étudier au préalable sur une feuille de papier, en décomposant bien le mouvement !

Et Olivier s'activa à dessiner sur une grande feuille de papier quadrillé une ribambelle de petits dessins. De loin, j'avais l'impression qu'il étudiait un dessin animé !... Je lui lançai :

P. — Alors Walt Disney en herbe, tu t'en sors de ton dérapage contrôlé ?

O. — Ça y est, j'ai terminé !... C'est passionnant à faire, mais quel boulot !... Voici le résultat de mes cogitations !... Tout d'abord, l'impact sur l'aile droite suivi d'un dérapage contrôlé :

```
 136 IF PEEK B=CODE "▪" THEN GOT
O 300
 300 PRINT AT 11,C;"*▪"
 310 PAUSE 25
 320 PRINT AT 10,C-1;"▪▪";AT 11
,C-1;"▪*▪";AT 12,C-1;"▪▪"
 330 PAUSE 25
 340 PRINT AT 10,C-2;"▪▪";AT 11
,C-2;"▪▪▪";AT 12,C;"▪▪"
```

O. — ... suivi, dans la foulée, par l'impact sur l'aile gauche, suivi du même dérapage contrôlé, mais cette fois dans l'autre sens :

```
 138 IF PEEK B+1=CODE "▪" THEN G
OTO 400
 400 PRINT AT 11,C;"▪*"
 410 PAUSE 25
 420 PRINT AT 10,C;"▪▪";AT 11,C
;"▪*▪";AT 12,C;"▪▪"
 430 PAUSE 25
 440 PRINT AT 10,C+1;"▪▪";AT 11
,C-1;"▪▪▪";AT 12,C-1;"▪▪"
```

O. — Tiens, c'est bizarre !... Le second programme que je viens d'écrire ne fonctionne pas du tout !

P. — Ne cherche pas trop longtemps !... Cela provient de la façon dont tu as écrit l'instruction PEEK. Sa syntaxe veut qu'il n'y ait qu'un seul nombre après. Comme tu y places une opération d'addition, il faut obligatoirement qu'elle soit encadrée par des parenthèses !

O. — Ah bon !... S'il n'y a que ça !... Je modifie aussitôt la ligne 138 en conséquence :

```
 138 IF PEEK (B+1)=CODE "▪" THEN
 GOTO 400
```

P. — Et alors ?

O. — Maintenant ça marche !

P. — Bien !... As-tu vérifié également si ton premier programme fonctionnait ?

O. — Non !... Un instant, ça va être vite fait !... Le temps de viser et c'est... pas bon !... Ça commence bien, mais ça finit mal !... Tu as une idée d'où ça provient ?

P. — Attends que je vérifie un peu ton programme !... Tout me paraît correct !... Ah, je viens de trouver !... Ce que tu constates, c'est l'enchaînement des deux programmes de dérapage. Tu commences avec celui de la ligne 3ØØ et tu continues après la ligne 34Ø avec celui de la ligne 4ØØ.

O. — Ah, c'est donc pour ça !... Et il y a moyen d'empêcher cela ?

P. — Oui !... Il te suffit de placer une instruction STOP entre la ligne 34Ø et la ligne 4ØØ !

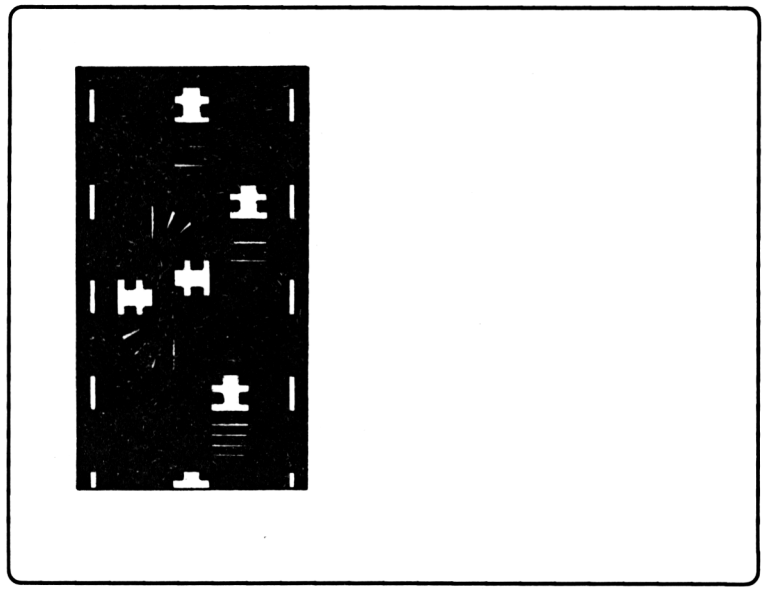
O. — C'est aussi simple que ça !... Je l'avais bien repérée cette instruction, mais je me demandais bien à quoi elle pouvait servir !... Maintenant, je sais !

P. — C'est vrai qu'à première vue, cette intruction STOP paraît absurde !... A quoi pourrait bien servir d'arrêter le déroulement d'un programme !... Eh bien maintenant, tu sais !... Elle sert à séparer entre elles certaines portions de programme !

O. — Ok !... Il ne me reste plus qu'à l'employer là où il faut :

```
23Ø STOP
35Ø STOP
```

P. — Voilà qui est fait !... On passe à autre chose ?



O. — Oui !... Je voudrais que tu me montres comment on peut se déplacer dans un labyrinthe sans traverser les murs !

Déplacement dans un labyrinthe

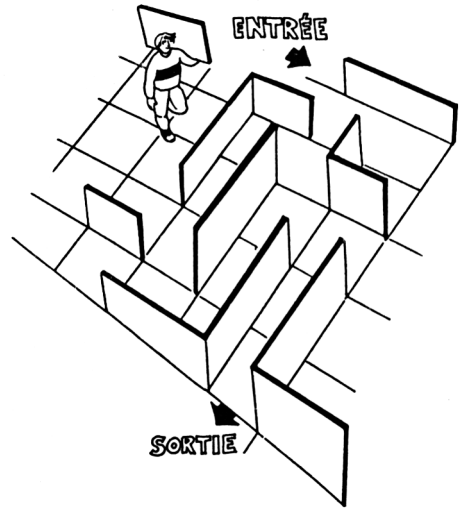
P. — C'est un sacré programme que tu me demandes là !... Heureusement que nous possédons la mémoire 16K, sinon je ne vois pas comment nous y arriverions !

O. — C'est vrai que cette fois-ci, je vais pouvoir dessiner un labyrinthe qui va occuper tout l'écran !... Ça, je peux dire que ça me fait plaisir !

P. — Dans ces conditions, je te laisse le dessiner comme il te plaît !

O. — Voilà, c'est fait !... Il ne me reste plus qu'à reprendre le programme du labyrinthe qu'on a déjà étudié ensemble il y a quelques semaines et à y rajouter quelques cloisons supplémentaires :

```
10 FOR L=0 TO 20 STEP 4
20 FOR C=0 TO 28 STEP 4
30 IF L=0 AND C<28 OR L=4 AND
C=20 OR L=8 AND C=4 OR L=8 AND C
=24 OR L=12 AND C=0 OR L=12 AND
C=20 OR L=16 AND C=4 OR L=20 AND
C<28 THEN GOSUB 200
40 IF C=0 AND L<20 OR C=4 AND
L=4 OR C=8 AND L=0 OR C=8 AND L=
12 OR C=12 AND L=4 OR C=12 AND L
=8 OR C=12 AND L=16 OR C=16 AND
L=0 OR C=16 AND L=8 OR C=16 AND
L=12 OR C=20 AND L=4 OR C=20 AND
 L=16 OR C=24 AND L=12 OR C=28 A
ND L<20 THEN GOSUB 250
50 NEXT C
60 NEXT L
```



P. — Comme tu dis, quelques cloisons supplémentaires...

O. — Bien !... Ceci étant écrit, il me reste encore les deux sous-programmes à faire !... Voici celui des cloisons horizontales :

```
200 FOR A=0 TO 4
210 PRINT AT L,C+A;"■"
220 NEXT A
230 RETURN
```

P. — Je vois que la mémoire fonctionne tout à fait correctement !

O. — La mémoire ?... De quelle mémoire parles-tu ?

P. — De la tienne, pardi !

O. — Ah !... Merci !... Pour la peine, voici l'autre sous-programme, celui des cloisons verticales :

```
250 FOR A=0 TO 4
260 PRINT AT L+A,C;"■"
270 NEXT A
280 RETURN
```

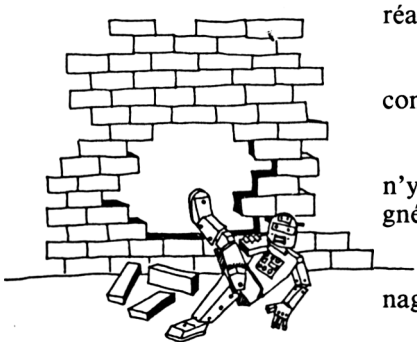
P. — Il est très bien agencé ton labyrinthe !... Mes compliments !

O. — Ça, c'était le plus facile !... Le reste me paraît nettement moins évident à programmer !

P. — Aie confiance !... N'oublie pas que nous sommes deux !

O. — C'est vrai !... Alors, voici le personnage que je veux déplacer au sein du labyrinthe !... Comme tu peux le deviner, il est fortement inspiré du jeu « Pacman » qui me plaît tant :

```
 70 LET X=14
 80 LET Y=10
 90 PRINT AT Y-1,X-1;"    ";AT Y
,X-1;"    ";AT Y+1,X-1;"    "
 100 IF INKEY$="5" AND X>2 THEN
LET X=X-1
 110 IF INKEY$="6" AND Y<18 THEN
 LET Y=Y+1
 120 IF INKEY$="7" AND Y>2 THEN
LET Y=Y-1
 130 IF INKEY$="8" AND X<26 THEN
 LET X=X+1
 140 PRINT AT Y-1,X-1;"▬▬";AT Y
,X-1;"▮▮";AT Y+1,X-1;"▬▬"
 150 GOTO 90
```



P. — Il est drôle, ton personnage !... Le fait qu'il clignote ne fait que rajouter à son réalisme !

O. — J'avoue ne pas être trop mécontent !... Toutefois, comme tu peux t'en rendre compte, ses déplacements présentent deux défauts importants !

P. — Effectivement !... D'abord, il peut passer à travers les murs et ensuite, si l'on n'y prend pas garde, il peut effacer tout un pan de mur quand il n'est pas exactement aligné dans l'axe d'un couloir !... Comme tu dis, deux beaux problème à résoudre !

O. — J'imagine que pour résoudre le premier, c'est-à-dire pour empêcher le personnage de traverser un mur, on doit pouvoir utiliser l'instruction PEEK !

P. — Tu as tout à fait raison !... On va employer cette instruction pour détecter si, deux carreaux en avant du personnage, il existe ou non un mur. Si l'on ne trouve rien, on l'autorisera à avancer d'une case. Si, par contre, on rencontre un mur, on interdira le moindre déplacement en direction de ce mur !

O. — Et comment on fait ça ?

P. — Comme pour les jeux précédents !... Suppose que tu veuilles déplacer ton personnage vers le haut de l'écran. Dans ce cas, il faut faire le test du mur deux lignes plus haut, c'est-à-dire 66 cases-mémoire en avant !

O. — Ah, j'ai compris !... Il faut d'abord que je définisse la position de mon personnage dans la mémoire d'affichage et qu'ensuite j'aille voir 66 cases avant s'il n'y a pas un mur !... C'est bien ça ?

P. — Exact !

O. — Alors, voici la ligne qui définit la position du personnage. Tu noteras que je la place avant les lignes d'instructions INKEY$, de façon à autoriser ou à interdire leur action :

```
95 LET B=1+X+33*Y+256*PEEK 163
97+PEEK 16396
```

P. — Très bien !

O. — Par contre, maintenant, je ne sais pas comment faire pour autoriser ou interdire la commande INKEY$!

P. — Je vais te montrer cela !... Dans le cas présent, il s'agit de la ligne 12Ø qui commande le déplacement de ton personnage vers le haut de l'écran, et qu'il te faut modifier ainsi :

```
120 IF INKEY$="7" AND Y>2 AND P
EEK (B-66)<>CODE "■" THEN LET Y=
Y-1
```

O. — Qu'est-ce que c'est que ce nouveau signe que tu viens d'employer ?... On dirait à la fois un mélange des signes « plus petit que » et « plus grand que » !

P. — C'est un signe qui veut dire « différent de » !... En fait, ce que j'ai rajouté dans cette ligne d'instructions, c'est-à-dire PEEK (B−66) <> CODE "■" permet d'aller vérifier si, deux lignes plus haut, il n'y a pas de mur qui fasse obstacle au déplacement vertical !

O. — Ça commence à devenir plus clair, pour moi !... Il faut, pour que Y = Y−1, que dans la case située deux lignes plus haut on détecte autre chose qu'un mur noir !... Et pour aller regarder cela deux lignes au-dessus, tu soustrais 66 cases-mémoire de l'adresse de la case-mémoire qu'occupe le personnage !... Qu'est-ce que c'est astucieux, tout ça !... Du coup, si j'ai bien compris, il suffit d'additionner 66 cases-mémoire à l'adresse B pour aller vérifier deux lignes plus bas !

P. — Tout à fait exact !... Comme il suffit d'enlever deux cases-mémoire pour le déplacement à gauche et d'en rajouter autant pour celui de droite !

O. — Dans le fond, c'est tout simple !... Je vais modifier en conséquence les autres lignes de commande :

```
100 IF INKEY$="5" AND X>2 AND P
EEK (B-2)<>CODE "■" THEN LET X=X
-1
110 IF INKEY$="6" AND Y<18 AND
PEEK (B+66)<>CODE "■" THEN LET Y
=Y+1
130 IF INKEY$="8" AND X<26 AND
PEEK (B+2)<>CODE "■" THEN LET X=
X+1
```

P. — Voilà une bonne chose de faite !... Maintenant, on va s'atteler au second problème, à savoir maintenir le déplacement du personnage dans le centre du couloir !

O. — Tu as une idée pour ça ?... Moi, je n'en ai pas !

P. — J'en ai une effectivement, bien que je ne la trouve pas géniale !

O. — Dis toujours !

P. — Cela consiste à vérifier, avant chaque déplacement, que le personnage est sur la ligne ou sur la colonne qui correspond au centre d'un couloir !

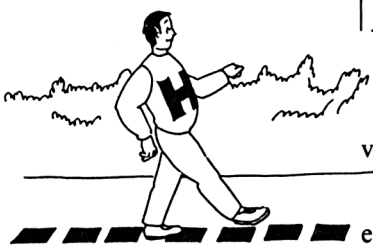
O. — Moi, je la trouve très bien cette idée !... Ainsi, si j'ai bien compris, il faut vérifier avant un déplacement horizontal par exemple, si le personnage est sur la ligne 2 ou 6 ou 10 ou 14 ou 18 !... C'est ça ?

P. — Tout à fait !... De même, pour un déplacement vertical, il faut vérifier si le personnage est sur la colonne 2 ou 6 ou 10 ou 14 ou 18 ou 22 ou 26 !

O. — Et ça s'écrit comment tout ça ?

P. — De diverses manières !... En voici une que je te propose :

```
 96 LET H=0
 97 LET V=0
 98 IF Y=2 OR Y=6 OR Y=10 OR Y=
14 OR Y=18 THEN LET H=1
 99 IF X=2 OR X=6 OR X=10 OR X=
14 OR X=18 OR X=22 OR X=26 THEN
LET V=1
```

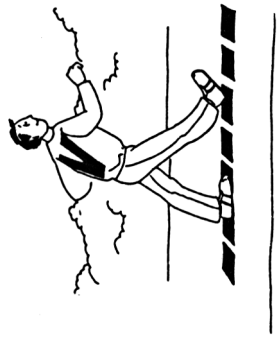


O. — Laisse-moi deviner tout seul comment ça marche !... Je suppose que H est la variable qui autorise ou non le déplacement horizontal ?

P. — Exact !... Quand elle vaut 0, le déplacement est interdit et quand elle vaut 1, il est autorisé !

O. — Et il n'est autorisé que si on est sur une ligne qui correspond au centre d'un couloir, c'est ça ?

P. — Eh oui !... Quant à la variable V, c'est pareil !

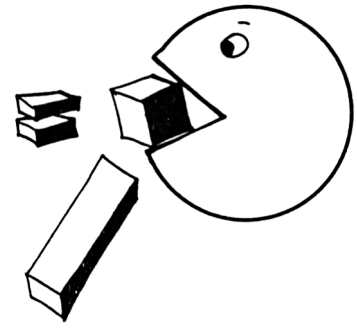


O. — Et pour rendre cette interdiction opérante, il suffit de rajouter l'une de ces deux variables dans chacune des lignes 100 à 130, en écrivant :

```
 100 IF INKEY$="5" AND X>2 AND P
EEK (B-2)<>CODE "■" AND H=1 THEN
 LET X=X-1
 110 IF INKEY$="6" AND Y<18 AND
PEEK (B+66)<>CODE "■" AND V=1 TH
EN LET Y=Y+1
 120 IF INKEY$="7" AND Y>2 AND P
EEK (B-66)<>CODE "■" AND V=1 THE
N LET Y=Y-1
 130 IF INKEY$="8" AND X<26 AND
PEEK (B+2)<>CODE "■" AND H=1 THE
N LET X=X+1
```

P. — Et voilà !... Tu vois que ce n'était pas si difficile que cela à imaginer un programme de déplacement dans un labyrinthe !

O. — Non, pas tant que ça !... D'ailleurs, tout ceci m'a donné quelques idées !... Je vais perfectionner ce programme pour en faire un jeu de « Pac-man » !... Il suffit que je rajoute quelques « nourritures » dans les couloirs que mon personnage va dévorer au passage !

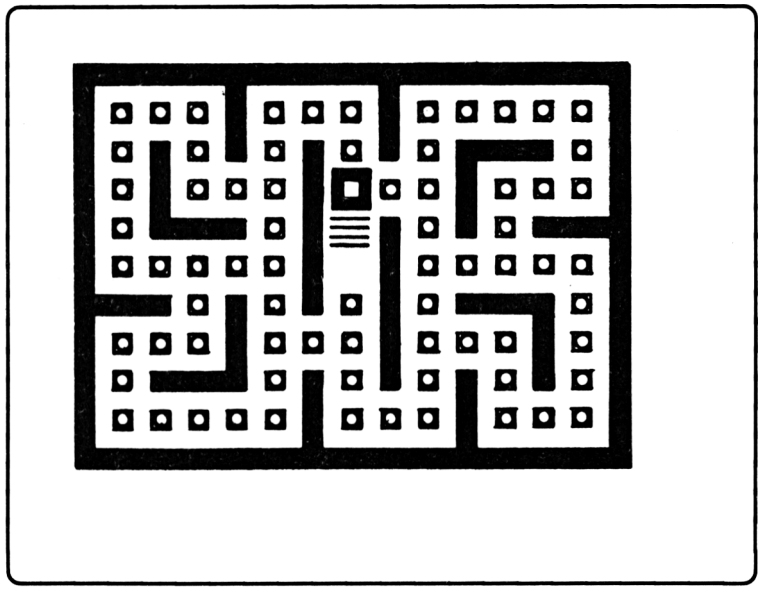


P. — J'imagine que, pour dessiner cela, tu n'as plus besoin de moi !

O. — Effectivement !... Je sais le faire tout seul !... Voici le programme en question :

```
61 FOR C=2 TO 26 STEP 4
62 FOR L=2 TO 18 STEP 4
63 PRINT AT L,C;"▣"
64 NEXT L
65 NEXT C
```

P. — Pas mal du tout !... J'imagine que maintenant, tu vas chercher à créer l'un de ces vilains fantômes qui va se déplacer au hasard dans le labyrinthe à la recherche de ton « affamé » !



O. — Tu as tout deviné !

Et c'est sur cette dernière réplique que j'abandonnai Olivier à son sort qui, soit dit entre nous, ne me paraissait pas triste du tout, absorbé qu'il était à inventer toutes sortes d'horribles créatures au destin étrange !... L'imagination tout de même, quand on lui donne des outils...

LA TOUCHE FINALE

Et la routine s'installait peu à peu, tranquillement, au fil des semaines qui s'écoulaient !... Nous en étions déjà à la septième !... Quant à l'ardeur d'Olivier, elle ne faiblissait pas, bien au contraire !... Il semblait éprouver une boulimie insatiable pour tout ce qui touchait de près ou de loin aux jeux vidéo et aux micro-ordinateurs domestiques !

Tout en réfléchissant sur ce que j'allais bien pouvoir lui apprendre aujourd'hui, ce qui en fait ne me causait guère de soucis vu que pratiquement c'était lui qui décidait de ce qu'il avait envie de découvrir, je mesurais le chemin parcouru depuis le début !... Il en était arrivé au point de pouvoir construire presque entièrement seul un jeu vidéo de son cru, bien à lui !... Il ne restait plus que quelques points de détail à voir ensemble. Bref, il en était aux finitions.

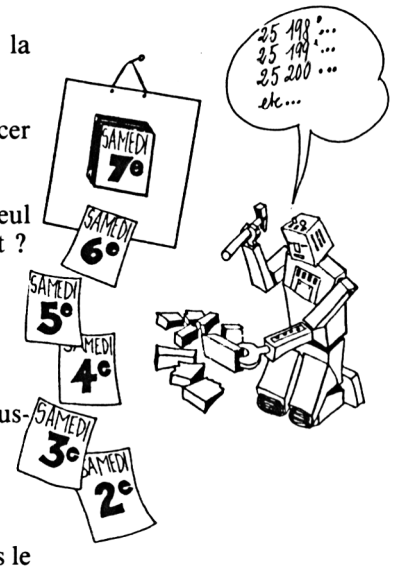
J'en étais à ce point de mes réflexions quand un tonitruant :

O. — Alors Papa, bien dormi ?... J'espère que tu es en forme parce que, moi, je te préviens, je le suis !

... me résonna dans les oreilles (surtout la droite...) ! Quand je vous disais que la journée s'annonçait intéressante...

P. — Eh bien, puisque tu es en forme, allons-y !... Par quoi veux-tu commencer cette journée ?

O. — Par l'affichage des résultats !... Depuis le début, on n'a pas encore fait un seul jeu dans lequel on ait eu à afficher des résultats !... Tu me montres comment on fait ?



L'affichage des résultats

P. — Bien sûr !... Maintenant, as-tu une idée du jeu qui pourrait nous servir à illustrer cela !

O. — Oh oui !... J'aimerais fabriquer un « casse-brique » !

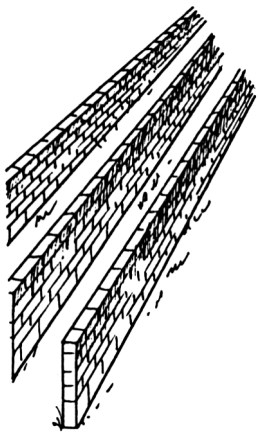
Quand je vous disais !... Les idées, c'est lui !... Moi, j'exécute !... Toutefois, dans le cas présent, j'avais un léger problème !... Je m'en ouvrais à lui :

P. — Pourrais-tu m'expliquer ce qu'est un « casse-brique », comme tu l'appelles ?

O. — Ah, parce que tu ne sais pas ce que c'est ! (là, je venais de descendre d'un cran dans son estime !) C'est un jeu dans lequel on doit casser des briques à l'aide d'une bille en acier. Les briques sont disposées sur plusieurs rangées auxquelles on ne peut accéder qu'à condition qu'il y ait déjà un trou dans la rangée précédente par où la balle peut passer. Bien entendu, plus la rangée est éloignée et plus le nombre de points que tu gagnes à chaque brique que tu casses est élevé !

P. — C'est tout un programme ce que tu viens de me décrire !... Une question encore : avec quoi contrôles-tu la bille ?

O. — Avec une raquette, tout simplement !



P. — Parfait !... J'ai donc tous les éléments en main pour t'aider à construire ce jeu !... Quant aux résultats que tu veux afficher, j'imagine qu'il s'agit du score que tu réalises ?

O. — Exactement !

P. — Alors, c'est à toi de jouer !... Construis le jeu du mieux que tu le peux, et je t'aiderai pour le score !

O. — Ok !... Voici donc l'aire de jeu de mon « casse-brique » :

```
 10 FOR C=0 TO 12
 20 PRINT AT 0,C;"▒"
 30 NEXT C
 40 FOR L=0 TO 21
 50 PRINT AT L,0;"▒";AT L,12;"▒
"
 60 NEXT L
```

P. — Jusque-là, tout va bien !... Je suppose qu'après, tu vas dessiner les murs de briques ?

O. — C'est ça !... Je vais d'ailleurs en dessiner trois, avec une ligne d'espacement entre chaque :

```
 70 FOR L=2 TO 6 STEP 2
 80 FOR C=1 TO 11
 90 PRINT AT L,C;"▬"
100 NEXT C
110 NEXT L
```

P. — Pourquoi laisses-tu une ligne d'espacement entre chaque mur ?

O. — Je n'en sais trop rien !... Parce que ça fait beau !

P. — Et bien moi, je vais t'expliquer pourquoi, sans le faire exprès, tu as choisi la disposition qui convenait !... Si tu prends une feuille de papier quadrillée et que tu y dessines ton aire de jeu, tu vas te rendre compte en mettant une croix dans les cases que traverse la bille, que seule une ligne sur deux est entièrement parcourue !

O. — Attends, je vérifie !... Ça y est, j'ai fini !... Effectivement, avec les rebonds en oblique et tout droit sur la raquette, seule une ligne sur deux peut être entièrement atteinte. Quelle chance !

P. — N'est-ce pas ?... Il ne reste plus qu'à placer ta raquette au bon niveau et ta chance sera totale !

O. — C'est vrai, ça !... Si je ne la place pas au bon niveau, je ne vais pouvoir détruire qu'une brique sur deux !... Là, je crois que ça va marcher comme ça :

```
 120 PRINT AT L,C;"    "
 130 IF INKEY$="5" AND C>1 THEN
LET C=C-1
 140 IF INKEY$="8" AND C<9 THEN
LET C=C+1
 150 PRINT AT L,C;"▬▬"
```

P. — Pas tout à fait !... Tu as oublié de définir les valeurs de L et de C !

O. — Là, je ne comprends pas !... Je devrais normalement recevoir un message d'erreur me signalant cet oubli !... Or, ce n'est pas le cas !... Pourquoi ?

P. — Parce que tu as déjà employé ces deux variables pour dessiner tes trois murs de briques. Et comme je t'ai déjà expliqué qu'après avoir exécuté une boucle du genre FOR... TO... NEXT, la variable conservait une valeur supérieure d'une unité à la valeur limite de la boucle, tu dois comprendre pourquoi ta raquette s'imprime en (7, 12), c'est-à-dire en plein sur ton mur extérieur !

O. — C'est donc pour ça !... Voilà, l'oubli est réparé :

```
111 LET L=20
112 LET C=5
```

P. — Il ne te reste plus qu'à refermer ta boucle de programme !

O. — Je suppose qu'il vaut mieux que je prévois large !... Est-ce que tu penses que 290 pourrait aller ?

P. — Je le crois !... En fait, d'après ce que tu m'as expliqué sur ce jeu, je pressens que tu auras besoin de plusieurs sous-programmes, ce qui revient à dire que ton programme principal sera relativement court !

O. — Alors, adjugé pour 290 :

```
290 GOTO 120
```

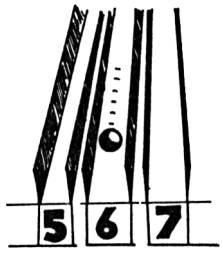
P. — Ok, c'est bon !... Ta raquette peut se déplacer à droite ou à gauche !... Que vas-tu écrire maintenant ?

O. — Le programme de déplacement et de rebond de la bille sur la raquette et les parois extérieures !... Voilà pour commencer le programme de déplacement :

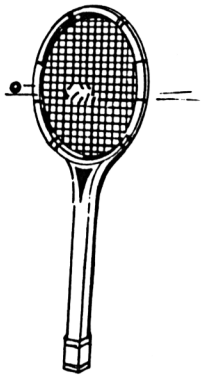
```
160 PRINT AT Y,X;"  "
170 LET X=X+U
180 LET Y=Y+V
190 PRINT AT Y,X;"."
```

O. — ... sans oublier les lignes de définition des variables X, Y, U et V comme je viens de le faire :

```
113 LET X=6
114 LET Y=8
115 LET U=0
116 LET V=1
```



P. — Parfait tout ça !... Je vois que tu as décidé d'engager la bille sur la colonne 6, de façon qu'elle vienne frapper en plein le centre de la raquette !



O. — Qu'elle traverse d'ailleurs !... Attends un peu que je lui écrive le programme de rebond, à celle-là :

```
200 IF Y=19 AND X>=C AND X<=C+2
THEN GOSUB 300
300 LET V=-V
310 IF X=C THEN LET U=-1
320 IF X=C+1 THEN LET U=0
330 IF X=C+2 THEN LET U=1
340 RETURN
```

P. — Voilà qui est fait !... Plus que les rebonds sur les parois extérieures à écrire et tu as terminé !

O. — Je m'y emploie !... Voici pour le mur du fond :

```
210 IF Y=1 THEN LET V=-V
```

O. — ... et voici pour les deux murs latéraux :

```
220 IF X=1 OR X=11 THEN LET U=-
U
```

P. — Bien !... Le plus gros est fait !... Te sens-tu d'attaque pour aborder la ligne de détection d'une brique ?

O. — Oui, oui !... Je ne dirais pas que c'est facile, mais je devrais pouvoir y arriver sans me tromper !... Il suffit de reprendre l'expression

IF PEEK 1 + X + 33 * Y + 256 * PEEK 16937 + PEEK 16396 = CODE « machin »

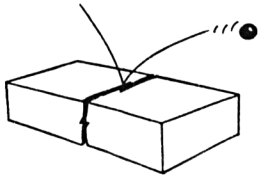
et ça doit marcher. Le problème, c'est de trouver où il faut la placer !

P. — Comme tu dis !... J'imagine que tu veux effacer la brique sur laquelle la bille rebondit ?

O. — Bien sûr !

P. — Alors, il n'y a qu'un endroit pour loger cette ligne d'instructions. Il te faut obligatoirement la placer entre la ligne 160 d'effacement de la bille et la ligne 190 de réimpression, après les deux lignes 170 et 180 de progression !

O. — Effectivement, cela ne laisse pas beaucoup de choix !... La voici donc cette fameuse ligne :



```
185 IF PEEK (1+X+33*Y+256*PEEK
16397+PEEK 16396)=CODE "▓" THEN
LET V=-V
```

P. — C'est bon !... La bille efface d'abord la brique puis rebondit !... C'est exactement cela que tu voulais obtenir, n'est-ce pas ?

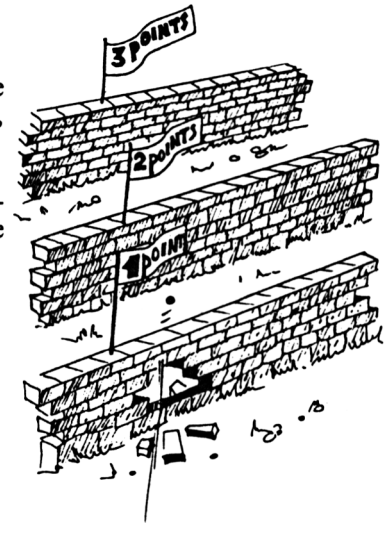
O. — Tout à fait !... Maintenant, il ne me reste plus qu'à comptabiliser les points et à les afficher !

P. — C'est donc à moi de jouer !... Cependant, avant de te montrer comment il faut procéder, j'aimerais que tu m'expliques en détail le nombre de points que tu souhaites attribuer à chaque brique détruite, si ce nombre varie selon les murs, etc...

O. — Oh, c'est tout simple !... Un point par brique du mur le plus proche de la raquette, deux pour le mur intermédiaire et trois pour le plus éloigné, c'est-à-dire le plus dur à atteindre !

P. — Cela me paraît très logique !... Donc, pour obtenir cela, je ne vois que la solution d'un sous-programme, sous-programme qui sera appelé par la ligne de détection des briques !... Voilà ce que je veux dire :

```
 185 IF PEEK (1+X+33*Y+256*PEEK
16397+PEEK 16396)=CODE "▄" THEN
GOSUB 350
 350 LET V=-V
 360 IF Y=6 THEN LET S=S+1
 370 IF Y=4 THEN LET S=S+2
 380 IF Y=2 THEN LET S=S+3
 390 RETURN
```



O. — J'avais compris !... Donc pour résumer, tu m'arrêtes si je me trompe, lorsque tu détectes une brique, tu vas voir dans le sous-programme si cela se passe à la ligne 6, 4, ou 2 et tu attribues 1, 2 ou 3 points selon le cas !... C'est bien cela ?

P. — Oui !... Quant à la variable S, elle correspond au nombre de points que tu totalises. Il ne faut donc pas oublier de la définir quelque part en début de programme. Je te propose donc de placer cette ligne en 117, comme ceci :

```
 117 LET S=0
```

O. — Tout ceci est très bien, mais ça ne me donne pas l'affichage des résultats !

P. — Tu as tout à fait raison !... Quand veux-tu connaître le nombre de points que tu totalises ?

O. — En permanence !... J'aimerais que, dans un coin de l'écran, ce total soit affiché en permanence !

P. — Qu'à cela ne tienne !... Voici ce que tu souhaites :

```
 385 PRINT AT 0,14;S
```

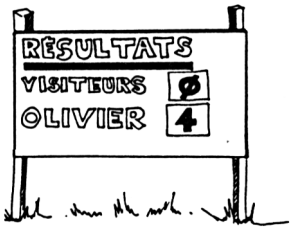
O. — Chouette, ça marche !... J'ai enfin l'affichage permanent de mon score !... Toutefois, ça me paraît un peu sec, ce nombre tout seul. Tu me montres comment on fait pour l'incorporer dans une phrase comme, voyons..., SCORE = S POINTS par exemple !

P. — Oui, bien sûr !... Pour obtenir cela, il faut modifier ta ligne d'instruction ainsi :

```
385 PRINT AT 0,14;"SCORE = ";S;
" POINTS"
```

P. — ... tu noteras que j'ai séparé par des points-virgules ce qui est placé entre guillemets et doit être imprimé tel que, de ce dont on veut connaître la valeur et qui n'est pas placé entre guillemets !

O. — C'est donc aussi simple que ça ?



P. — Eh oui !... N'oublie toutefois pas, quand tu écris ce genre de ligne d'instructions, de prévoir les espaces entre les mots et les chiffres !

O. — Maintenant, je comprends mieux pourquoi tu as écrit le mot POINTS en démarrant par un espace !... Mais dis-moi, cela ralentit drôlement le jeu cet affichage !

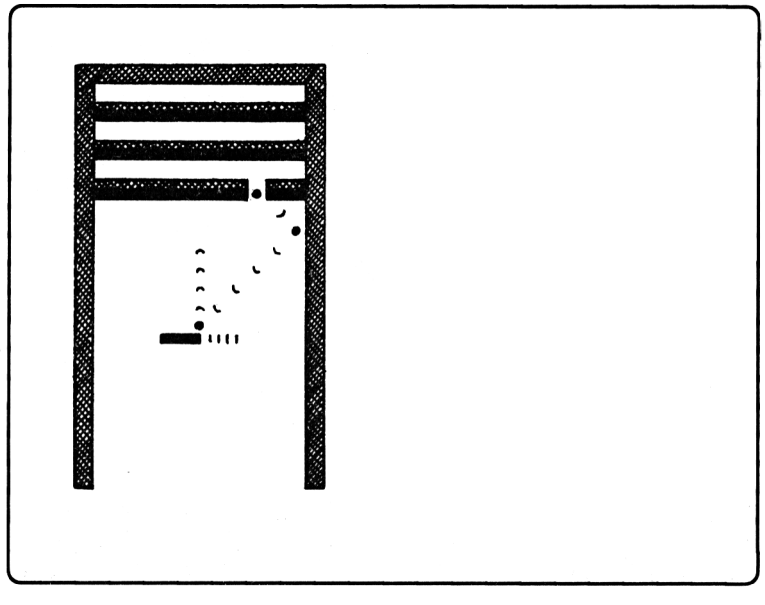
P. — N'est-ce pas ?... Je t'ai cependant laissé procéder à ta guise pour qu'ensuite tu puisses mieux comprendre pourquoi il vaut mieux ne pas l'effectuer en cours de jeu, mais plutôt à la fin !

O. — Quand je rate la balle, par exemple ?

P. — Oui !... Il te suffit d'écrire pour cela :

```
230 IF Y=21 THEN GOTO 400
385
400 PRINT AT 0,14;"SCORE = ";S;
" POINTS"
410 STOP
```

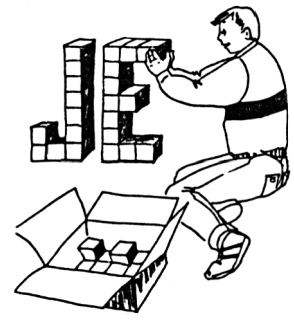
O. — Et j'ai mon résultat en fin de partie !... C'est super comme ça !... Dommage qu'on ne puisse pas accélérer la balle !



P. — On ne le peut pas, mais tu peux par contre rapprocher la raquette !... Essaye donc avec L égal à 14. Tu verras, c'est nettement mieux !

O. — Ok !... Du coup, il faut que je modifie les lignes 111 et 200 ainsi :

```
111 LET L=14
200 IF Y=13 AND X>=C AND X<=C+2
THEN GOSUB 300
```



P. — Et voilà !... J'espère que tu es content de ton premier jeu vidéo entièrement terminé ?

O. — Et comment !... D'autant plus que c'est moi qui l'ai construit tout seul !

P. — Bien !... Puisqu'il en est ainsi, on va pouvoir passer à la suite, j'ai nommé la remise en jeu automatique !

La remise en jeu automatique

O. — Qu'entends-tu exactement par là ?

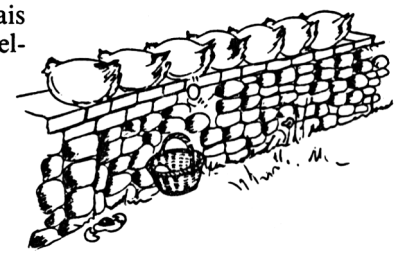
P. — Je te répondrais bien comme Pierre Dac, mais en l'occurence, il s'agit d'effectuer la comptabilité du nombre de parties auxquelles tu as le droit de jouer !

O. — Un peu comme le nombre de balles dans une partie de golf miniature ?

P. — C'est cela !... Maintenant, as-tu une idée du jeu qui nous permettrait d'illustrer ce mécanisme de remise en jeu ?

O. — Non, pas vraiment !

P. — Moi, j'en ai bien une, mais je ne suis pas sûr qu'elle te plaise !... Je dirais même qu'il s'agit d'un jeu un peu « loufoque sur les bords » auquel je pense depuis quelques jours !



O. — Bah, dis toujours !... J'en ai entendu d'autres !

P. — C'est le jeu des poules pondeuses !... Tu as sept poules sur un mur ...

O. — ... qui picoraient du pain dur !... Tu vois, je connais mes classiques !

P. — Je disais donc qu'il y avait sept poules sur un mur qui pondaient des œufs à tour de rôle. Toi, tu disposes d'un panier qu'il faut placer en dessous de la poule qui vient de pondre, de façon à récolter son œuf. Bien sûr, si tu le rates ou s'il touche le bord du panier, il s'écrase !

O. — Beurrkk !

P. — Attends, je n'ai pas fini !... Si nous sommes doués, on peut tenter de récolter trois œufs dans le même panier, qu'il faudrait placer côte à côte bien entendu !... Si un œuf tombe sur un autre œuf, les deux se cassent comme il se doit !

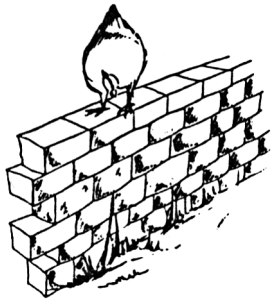
O. — Tout compte fait, il est « marrant » ton jeu !... Il me rappelle un certain KABOUM de chez...

P. — Chut, pas de marque ici !

O. — Par contre, je ne vois pas très bien où se fait la remise en jeu automatique dans tout ça ?

P. — Elle intervient dans la mesure ou un nouvel œuf sera pondu tant que tu n'en auras pas trois dans ton panier. Après, il suffira de comptabiliser les œufs pondus pour juger de tes qualités de ramasseur d'œufs.

O. — Ok, ça me « botte » !... Toutefois, si tu veux bien, je préférerais que ce soit toi qui commences, puisque l'idée du jeu, c'est toi qui l'as en tête !



P. — Comme tu voudras !... Voici donc le mur :

```
10 FOR A=1 TO 29 STEP 4
20 PRINT AT 3,A;"▒▒▒▒"
30 NEXT A
```

O. — Il est « marrant » ton mur, il est tout troué !

P. — C'est pour mieux laisser passer les œufs, mon enfant !

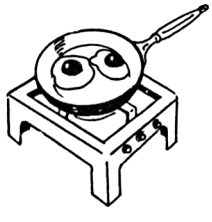
O. — Tu m'en diras tant !... Je suppose que tu vas placer les poules juste au-dessus de chaque trou ?

P. — Tu supposes juste !... Les voici d'ailleurs qui se mettent en place toutes les sept :

```
40 FOR A=3 TO 27 STEP 4
50 PRINT AT 0,A;"▄▄▄▄";AT 1,A;"
▄▄▄▄";AT 2,A;"▌ ▌"
60 NEXT A
```

O. — Elles ne sont pas tristes, tes poules !

P. — Attends de les voir pondre leurs œufs !... Tiens, toi qui ne fais rien, écris-moi donc le programme qui fasse surgir un œuf du derrière d'une poule, prise au hasard et fais-le ensuite chuter vers le sol !



O. — Tout de suite, chef !... Je vais te mijoter ça aux petits oignons !... Et un œuf qui marche, un !

```
70 LET N=INT (RND*7+1)
80 LET L=1
90 IF L>1 THEN PRINT AT L,4*N;
" "
100 LET L=L+1
105 IF L=22 THEN GOTO 70
110 PRINT AT L,4*N;"O"
190 GOTO 90
```

P. — Pas mal !... Je vois que tu as pensé à interdire l'effacement de l'œuf tant que sa variable de descente valait 1, de façon à éviter de faire un trou dans les poules !

O. — Les pauvres !... J'allais quand même pas leur faire ça !...

P. — Bien !... Maintenant que nous avons les œufs, il nous faut un panier !... Ça, c'est encore pour toi !

O. — Je m'en serais douté !... Je suppose, que d'après l'énoncé que tu m'as fait du jeu tout à l'heure, il te faut un panier à trois cases qui puisse contenir trois œufs ?

P. — Eh oui !

O. — Bon !... Eh bien, le voici :

```
 120 PRINT AT 20,X;"      ";AT 21
,X;"      "
 130 IF INKEY$="5" AND X>1 THEN
LET X=X-1
 140 IF INKEY$="8" AND X<27 THEN
 LET X=X+1
 150 PRINT AT 20,X;"▮    ▮";AT 21
,X;" ▬▬▬▬ "
```

P. — Tu n'aurais rien oublié, par hasard ?

O. — Ah si, c'est vrai !... Décidément, c'est ma manie ça !... Voici la ligne d'affectation de la variable X !... Au fait, où faut-il placer cette ligne ?

P. — N'importe où, pourvu que cela soit avant la ligne 70 !

O. — Je vais donc la mettre en 65 :

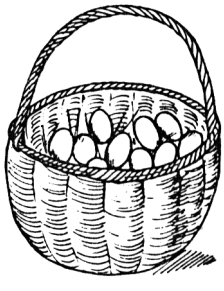
```
 65 LET X=14
```

P. — Bien !... Maintenant, on va écrire la ligne de détection de la rencontre d'un œuf avec le fond du panier !

O. — Est-ce que tu fais ça avec l'instruction PEEK ?

P. — Non car, dans notre cas, nous connaissons les coordonnées de l'œuf (L, N*4) et du fond du panier (20, X+1 à X+3) ! Il suffit donc d'écrire :

```
 160 IF L=20 AND N*4>=X+1 AND N*
4<=X+3 THEN GOTO 200
```



O. — C'est vrai !... C'est le programme d'impact tout simple !... Toutefois, le plus dur n'est pas fait. Il faut maintenant trouver un moyen pour conserver l'œuf dans le panier une fois qu'il a atterri dedans !

P. — Ce n'est effectivement pas évident !... Attends que je réfléchisse à cela !... Ça y est, je viens de trouver une solution.. Normalement, elle devrait convenir !

O. — Tu m'expliques !

P. — Voilà !... Le fond du panier comprend trois cases que l'on va appeler respectivement A, B et C. Au départ, on va affecter à ces trois variables la valeur 0 qui correspond au numéro de code de l'espace, et on va modifier ainsi la ligne 150 d'écriture du panier :

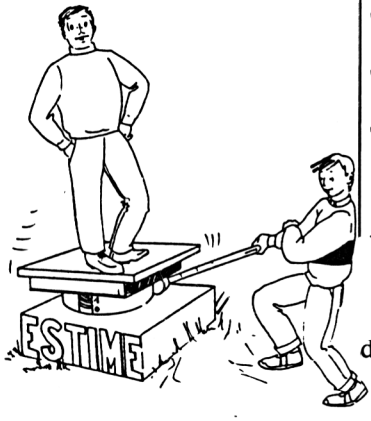
```
 61 LET A=CODE " "
 62 LET B=CODE " "
 63 LET C=CODE " "
```

```
150 PRINT AT 20,X;"▮";CHR$ A;CH
R$ B;CHR$ C;"▮";AT 21,X;"▬▬▬"
```

O. — Je crois que je devine la suite !... Lorsqu'un œuf tombe dans une des cases du panier, tu attribues à la variable correspondante le numéro de code de la lettre O qui symbolise l'œuf !... C'est ça, hein ?

P. — Exactement !... D'ailleurs, voici le sous-programme qui réalise cela :

```
200 IF N*4=X+1 THEN LET A=CODE
"O"
210 IF N*4=X+2 THEN LET B=CODE
"O"
220 IF N*4=X+3 THEN LET C=CODE
"O"
230 GOTO 70
```



O. — C'est super, ça marche !... Bravo Papa ! (là, je venais de remonter d'un cran dans son estime !).

P. — Malgré tout, cela ne me convient pas encore !... As-tu vu comme l'œuf s'efface, en arrivant dans le panier, pour réapparaître ensuite !... J'aimerais pouvoir supprimer ce défaut !

O. — Je crois que j'ai trouvé d'où ça provient !... C'est parce qu'on a écrit les lignes d'instructions de la chute des œufs avant celles du déplacement du panier !... Je crois que si on les inversait, tout irait bien !

P. — Il me semble que tu as raison !... Je vais essayer tout de suite :

```
82 PRINT AT 20,X;"▬▬▬";AT 21
,X;"      "
84 IF INKEY$="5" AND X>1 THEN
LET X=X-1
86 IF INKEY$="8" AND X<27 THEN
LET X=X+1
88 PRINT AT 20,X;"▮";CHR$ A;CH
R$ B;CHR$ C;"▮";AT 21,X;"▬▬▬"
120
130
140
150
190 GOTO 82
```

O. — Tu vois, j'avais raison, c'est ça qu'il fallait faire !

P. — Effectivement !... En y réfléchissant, on peut se rendre compte que l'ancienne ligne 150 effaçait obligatoirement l'œuf en redessinant le panier !... Il fallait donc imprimer l'œuf après le panier !

O. — Puisque ça marche, si on passait au programme d'écrasement des œufs !

P. — Tu as raison, d'autant plus qu'il y en a trois : lorsque l'œuf tombe sur le bord du panier, sur un autre œuf ou par terre !

O. — Il y a du travail en perspective !... Par lequel on commence ?

P. — Par le plus simple, c'est-à-dire par l'écrasement au sol.

O. — Ok !... C'est comme si c'était fait !... Je vais d'abord commencer par dessiner le sol :

```
1 FOR C=Ø TO 31
2 PRINT AT 21,C;"▬"
3 NEXT C
```

O. — Petit problème !... Le panier efface le sol !... C'est gênant !... Je vais donc le remonter d'au moins trois carreaux !... Ça m'oblige à modifier les lignes 82, 88 et 16Ø :

```
82 PRINT AT 17,X;"      ";AT 18
,X;"      "
88 PRINT AT 17,X;"▮";CHR$ A;CH
R$ B;CHR$ C;"▮";AT 18,X;"▬▬▬"
16Ø IF L=17 AND N*4>=X+1 AND N*
4<=X+3 THEN GOTO 2ØØ
```

O. — Voilà un problème réglé !... Il me reste à écrire la ligne de détection de l'impact au sol :

```
17Ø IF L=2Ø THEN GOTO 24Ø
```

O. — ... suivie du programme d'écrasement de l'œuf !... Quel gaspillage tout de même...

```
24Ø PRINT AT L,N*4-1;"(  )"
25Ø PAUSE 12
26Ø PRINT AT L-1,N*4-1;"▚ ▞";AT
L,N*4-1;"    "
27Ø PAUSE 12
28Ø PRINT AT L-1,N*4-1;"    "
29Ø GOTO 7Ø
```

P. — Très bonne idée que celle d'employer des parenthèses !... On dirait que l'œuf se casse en deux !

O. — N'est-ce pas !... C'est exactement l'effet que je voulais produire !

P. — Pendant que tu tapais ce programme j'ai réfléchi aux différents programmes d'impact qu'il nous fallait élaborer !... En fait, on va pouvoir se servir du même programme pour l'écrasement au sol et sur le bord du panier !

O. — Tant mieux !... Il me suffit donc d'écrire la ligne de détection de l'impact sur le bord du panier, et de renvoyer vers le même sous-programme que précédemment :

```
180 IF L=16 AND 4*N=X OR L=16 A
ND 4*N=X+4 THEN GOTO 240
```

P. — Et ça marche !... Bravo !... Maintenant, il nous reste à aborder l'exercice de style, à savoir l'impact d'un œuf sur un autre !

O. — Ça ne va pas être facile, facile !... Comment vois-tu la chose ?

P. — Je la vois ainsi : lorsqu'un œuf arrive dans le fond du panier, la ligne 160 fait appel au sous-programme 200. Or, ce sous-programme attribue bêtement la valeur CODE "O" à la variable qui correspond à l'emplacement de l'œuf dans le panier. Ce qu'il faut rajouter, c'est un test préalable de la valeur de cette variable. Si elle vaut CODE " ", c'est que la place est libre et on peut donc lui attribuer la valeur CODE "O". Si, par contre, elle vaut déjà CODE "O", c'est qu'il y a déjà un autre œuf et dans ce cas on renvoie sur le programme d'écrasement de l'œuf !

O. — Je crois que j'ai compris !... Tu me montres un exemple !

P. — Oui !... Je vais t'écrire les deux lignes qui vont correspondre à la variable A et toi, tu feras les deux autres :

```
200 IF N*4=X+1 AND A=CODE "O" T
HEN GOTO 300
205 IF N*4=X+1 AND A=CODE " " T
HEN LET A=CODE "O"
```

O. — Ok !... A moi de jouer !

P. — Attends encore un peu !... Je viens de me rendre compte qu'il vaut mieux que je t'écrive également le programme d'écrasement, car je vois poindre à l'horizon une petite complication qui, si on n'y prend pas garde, t'obligerait à tout réécrire !

O. — Merci d'y avoir pensé !

P. — Voici donc le programme d'écrasement !... Tu noteras qu'au début je doit remettre la valeur de la variable à 0 et, pour cela, je vais devoir un peu jongler avec l'instruction GOTO :

```
300 LET A=CODE " "
310 GOTO 360
360 PRINT AT L,N*4;"▒"
370 PAUSE 12
380 PRINT AT L-1,N*4-1;"▚ ▞";AT
L,N*4;" "
390 PAUSE 12
400 PRINT AT L-1,N*4-1;"    "
410 GOTO 70
```

O. — Ah, je crois comprendre !... Tu m'as laissé quatre lignes d'instruction entre 310 et 360 pour l'affectation de la valeur CODE " " aux deux autres variables B et C !... Voici donc la suite :

```
210 IF N*4=X+2 AND B=CODE "O" T
HEN GOTO 320
215 IF N*4=X+2 AND B=CODE " " T
HEN LET B=CODE "O"
220 IF N*4=X+3 AND C=CODE "O" T
HEN GOTO 340
225 IF N*4=X+3 AND C=CODE " " T
HEN LET C=CODE "O"
320 LET B=CODE " "
330 GOTO 360
340 LET C=CODE " "
350 GOTO 360
```

P. — Il n'y a rien qui te choque ?

O. — Non, je ne vois pas !

P. — Regarde ce que tu as écrit à la ligne 350 !

O. — Ben, GOTO 360 !... Qu'est-ce qu'il y a de drôle ?

P. — La 360, c'est la ligne suivante !

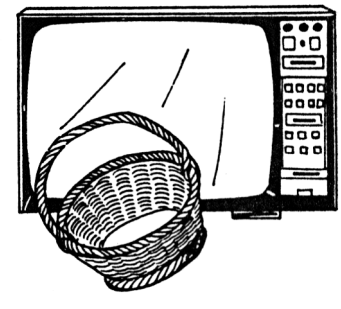
O. — Tiens, c'est vrai !... Mais alors, elle ne sert à rien cette ligne ?

P. — Je ne te le fais pas dire !

O. — Pour en revenir au jeu, il n'est pas assez rapide pour être intéressant !... Il faudrait pouvoir accélérer la descente de l'œuf...

P. — ... ou le déplacement du panier !... Je viens d'avoir une idée terrible pour accélérer le mouvement du panier !... Ecris cela et dis-moi ce que tu en penses :

```
81 LET U=0
83 IF INKEY$="5" AND U>-3 THEN
LET U=U-1
84 IF INKEY$="8" AND U<3 THEN
LET U=U+1
85 LET X=X+U
```



O. — Pas mal !... Tu as transformé les touches 5 et 8 en frein et accélérateur !... Il n'y a qu'un défaut, le panier peut sortir de l'écran !

P. — Qu'à cela ne tienne !... Deux petites butées et le tour est joué :

```
86 IF X<0 THEN LET X=0
87 IF X>27 THEN LET X=27
```

O. — C'est mieux !... Il ne te reste plus qu'à me montrer ce dont tu m'avais parlé, à savoir la remise en jeu automatique !

P. — En fait, elle existe déjà comme tu as pu le constater puisque chaque sous-programme, qu'il soit d'impact ou de présence au fond du panier, renvoie systématiquement sur la ligne 70 avec un GOTO !... Ce qu'il faut, c'est limiter le nombre de parties !

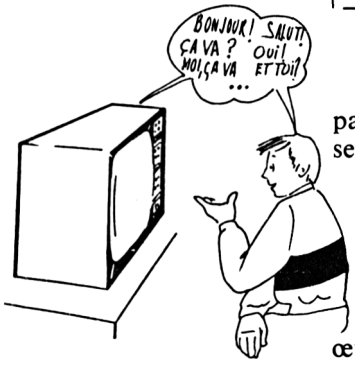
O. — Et ça se fait comment ?

P. — En affectant une variable à cela !... Appelons-la P par exemple !... Maintenant, combien de parties penses-tu qu'il soit raisonnable de jouer pour avoir quelques chances de terminer avec trois œufs dans le panier ?

O. — Je ne sais pas !... Sept ?

P. — Va pour sept !... On peut donc écrire la chose ainsi, en ayant au préalable dénuméroté la ligne 7Ø à 72 :

```
 69 LET P=Ø
 7Ø LET P=P+1
 71 IF P=8 THEN GOTO 43Ø
 72 LET N= INT (RND*7+1)
 43Ø PRINT AT 1Ø,Ø;"VOUS AVEZ PE
RDU."
 44Ø STOP
```



O. — C'est « marrant », cette façon qu'a la machine de me répondre !... Je n'aurais pas écrit le programme moi-même, j'aurais l'impression que c'est elle qui invente les phrases toute seule !

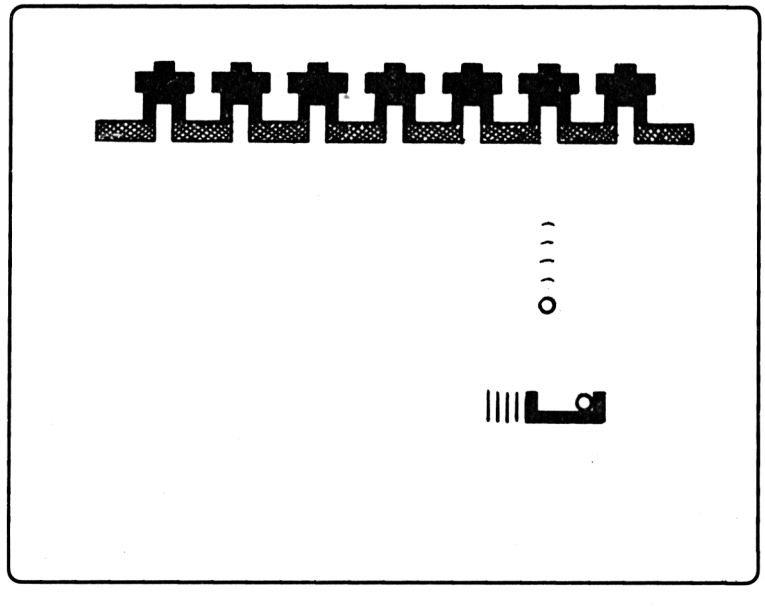
P. — C'est ça le miracle de l'informatique !

O. — J'imagine qu'on peut également lui faire dire qu'on a gagné ?

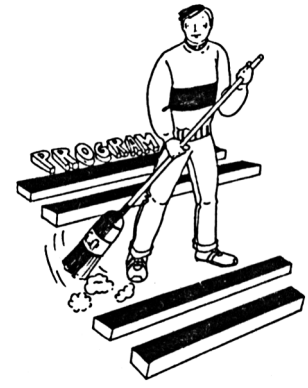
P. — Bien sûr !... Pour cela, il faut écrire la ligne qui détecte la présence des trois œufs dans le panier et afficher le nombre d'œufs cassés :

```
 227 IF A=CODE "O" AND B=CODE "O
" AND C=CODE "O" THEN GOTO 45Ø
 45Ø PRINT AT 1Ø,Ø;"BRAVO , VOUS
N AVEZ CASSE QUE ";P-3;" OEUFS.
"
```

O. — Génial !... Cette machine est géniale !



P. — Elle l'est peut-être, je t'en laisse seul juge, mais n'oublie pas dans cette affaire le rôle au moins aussi important de l'opérateur !... Pour t'imager cela, je pourrais te dire que pour arriver premier dans une course automobile, il faut avoir une bonne machine et savoir conduire !... En informatique, c'est pareil !



O. — C'est vrai !... Puisque le jeu est terminé, je vais en profiter pour le remettre un peu en ordre !

P. — Bonne idée !... Il y a une ou deux lignes inutiles que tu peux supprimer, telle que la 1Ø5 ou la 35Ø !

O. — Ça, c'est déjà fait !... Je pensais plutôt à tout renuméroter et, aussi, à remplacer le 4 *N qui revient très souvent par N tout simplement, en modifiant la ligne 72 ainsi :

```
72 LET N=4*INT (RND*7+1)
```

P. — Très bonne initiative !... Pendant que tu fais cela, je vais réfléchir à ce que nous allons étudier après !

Et pendant qu'Olivier « pianotait » sur son clavier, je passais en revue les différentes notions et instructions qu'il nous restait encore à voir ensemble. Je jetais mon dévolu sur les chaînes de caractères. Cependant, quel allait être le jeu qui nous servirait à illustrer cette notion ?

Les chaînes de caractères

La solution ne se fit pas attendre !... Elle vint, comme prévu, sous la forme d'une réflexion anodine que me fit Olivier lorsqu'il eut terminé la mise à jour de son programme :

O. — Pendant que je renumérotais toutes ces lignes, je me disais que ça serait chouette si l'ordinateur pouvait nous commenter une course de chevaux, à la manière de Léon Zitrone !

P. — Pourquoi pas ?... Et même, je vais profiter de cette occasion pour te parler des chaînes de caractères !

O. — Les chaînes de caractères ?... C'est quoi ?

P. — Je t'expliquerai cela au cours de l'élaboration de cette course de chevaux !

O. — Bon, bon !... Comme tu voudras !... Je suppose qu'il va falloir te dessiner des chevaux ?

P. — Cela me paraît indispensable !

O. — Alors, les voilà tes « pur sang » :

```
10 FOR N=1 TO 5
20 PRINT AT 4*N-4,0;" ▄   O"
30 PRINT AT 4*N-3,0;" ██"
40 PRINT AT 4*N-2,0;" ▌ ▌"
50 NEXT N
```

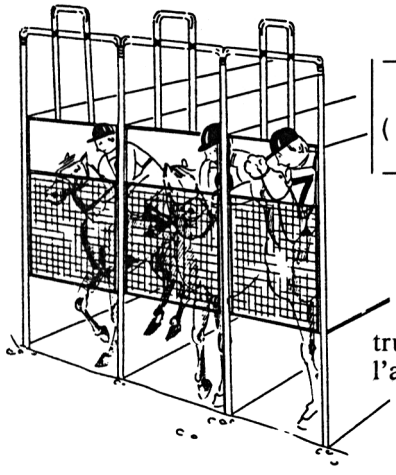
P. — Ils sont drôles !... Ils ressemblent un peu à des toutous !

O. — Tu sais, ce n'est pas évident de représenter un cheval avec des carrés !

P. — Je le sais !... Si je t'ai fait cette remarque, c'est pour te blaguer !... Ce qu'il nous manque, c'est la haute résolution graphique !... Seulement ça, c'est une autre histoire !... En attendant, ne pourrais-tu pas mettre un dossard à chacun des chevaux ?

O. — Et tu ferais ça comment, toi ?

P. — En réécrivant ta ligne 3Ø ainsi :



```
30 PRINT AT 4*N-3,0;"  ";CHR$
(156+N);"  "
```

O. — A quoi correspond le nombre 156 ?

P. — Au code du chiffre Ø en vidéo inversée !... Ainsi, si par exemple N vaut 1, l'instruction CHR$ (156 + 1) te fera apparaître un chiffre 1 en vidéo inversée à la suite de l'arrière-train du cheval !

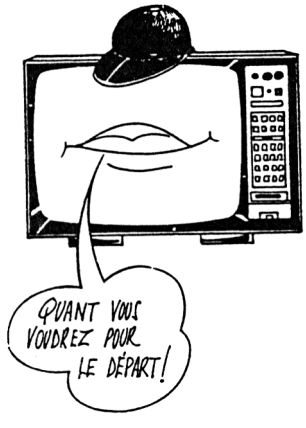
O. — Astucieux !... Faudra que je me souvienne de cette méthode !

P. — Et si on passait à la suite ?... Il nous manque une ligne de départ et une ligne d'arrivée !... Tu nous les dessines ?

O. — C'est comme si c'était fait :

```
60 FOR L=0 TO 18
70 PRINT AT L,5;"  ";AT L,31;"  
"
80 NEXT L
```

P. — Bien !... Maintenant, on va faire « parler » la machine !... Je te propose de lui faire prononcer une phrase dans le genre : « Les chevaux sont sous vos ordres. Quand vous voudrez pour le départ » !



O. — Super !... Tu me laisses l'écrire ?

P. — Bien sûr !... Toutefois, fais attention à la syntaxe d'écriture, car il y a pas mal de pièges !

O. — Ok !... La voici donc cette phrase !... Tout du moins la première partie :

```
90 PRINT AT 20,0;"LES CHEVAUX
SONT SOUS VOS ORDRES."
```

P. — As-tu remarqué le point qui apparaît tout seul sur la ligne suivante ?

O. — Ne m'en parle pas !... Elle est peut-être géniale cette machine mais, parfois, qu'est-ce qu'elle « tient comme couche » !... Je mets le point pour faire bien et elle me le place tout seul sur une ligne !... Je vais de ce pas l'effacer !

P. — En même temps que tu feras cela, supprime donc également le AT 2Ø, Ø ; et dis-moi ce qui se passe !

O. — Comme tu veux, mais je ne vois pas pourquoi tu veux écrire cela en haut de l'écran !... Enfin, puisque tu insistes :

```
90 PRINT "LES CHEVAUX SONT SOUS VOS ORDRES"
```

P. — Et alors ?

O. — C'est drôle, ça ne s'inscrit pas en haut de l'écran comme je le pensais, mais juste en dessous des chevaux, c'est-à-dire sur la ligne 19 !... Pourquoi ça fait ça ?

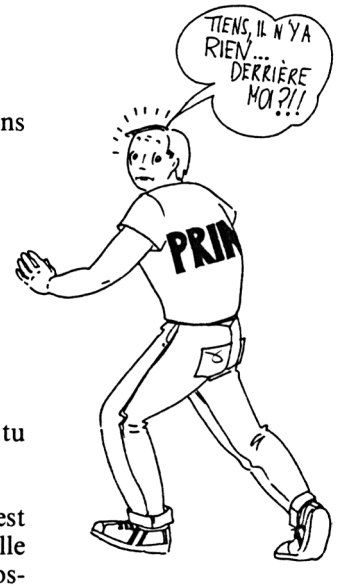
P. — La raison en est que quand tu ne spécifies pas après une instruction PRINT la ligne de l'écran sur laquelle l'inscription doit être exécutée, l'ordinateur décide automatiquement de placer celle-ci à la suite de la dernière ligne utilisée. Or, dans notre cas, il s'agit de la ligne 18 sur laquelle s'est terminé le dessin des deux lignes de départ et d'arrivée. C'est pourquoi tu retrouves ton texte en ligne 19 !

O. — Ah bon !... L'ennui, c'est que c'est à la ligne 20 que je le veux !

P. — C'est possible !... Il suffit pour cela de rajouter une instruction PRINT sans rien à sa suite !

O. — Quelque chose comme :

```
85 PRINT
```



P. — Oui !... Tout à fait !... Ainsi, une ligne vide s'inscrit en 19 et le texte que tu veux à la suivante !

O. — Encore une façon de programmer des espaces, que je découvre !... Bon, c'est pas tout ça, à la seconde partie de ma phrase !... J'aimerais qu'elle prenne la place de celle que je viens d'écrire !... Or, si j'ai bien compris ce que tu m'as expliqué, ce n'est pas possible autrement qu'en employant l'instruction AT !

P. — Exact !... Dans ce cas, il te faut remonter, et pour cela, je ne vois pas d'autre solution que celle que tu viens de trouver !

O. — Alors, voici la suite :

```
100 PRINT AT 20,0;"TAPEZ LE MOT "DEPART" ET APPUYEZ ENSUITE SUR NEWLINE."
```

O. — ... Tiens, j'ai fait une erreur de syntaxe !... Je me demande bien où ?

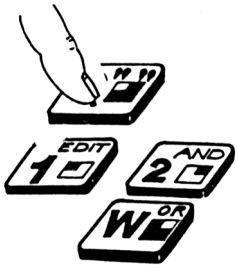
P. — Je vais te le dire !... Tu as placé des guillements de part et d'autre du mot DEPART et c'est cela que la machine refuse. Elle ne comprend plus rien si tu lui mets des guillemets entre les guillemets de l'instruction PRINT !

O. — Alors, il n'est pas possible de mettre des guillemets dans une phrase ?

P. — Si fait, mais il faut employer pour cela les autres guillemets, ceux qui sont sur la touche Q !

O. — C'est donc à ça qu'ils servent !... Je m'étais déjà interrogé à leur sujet et je n'avais pas trouvé de réponse !... Le temps de les changer dans la phrase et tout va rentrer dans l'ordre :

```
100 PRINT AT 20,0;"TAPEZ LE MOT
""DEPART"" ET APPUYEZ ENSUITE S
UR NEWLINE."
```



O. — ... enfin presque !... Je ne veux pas des guillemets doubles !... Ça ne ressemble à rien !

P. — Ne t'indigne pas trop vite !... Ils n'apparaissent en double que dans ton programme, pour bien les distinguer des autres. Par contre, sur l'écran, ils apparaissent comme des guillemets ordinaires !

O. — C'est vrai !... Que de subtilités dans ce BASIC !

P. — Encore un petit truc et j'en aurai fini !... As-tu remarqué que le mot ENSUITE ne débutait pas au même niveau que le mot TAPEZ, mais qu'il était décalé d'un espace vers la droite ?

O. — Oui !... J'imagine qu'il suffit de supprimer l'espace entre ce mot et le précédent pour remédier à cela ?

P. — Tout à fait !... De même, il t'arrivera de devoir rajouter plusieurs espaces entre deux mots, de façon à éviter que le second ne soit coupé en bout de ligne !

O. — Quelle cuisine !... Voici donc la phrase réécrite :

```
100 PRINT AT 20,0;"TAPEZ LE MOT
""DEPART"" ET APPUYEZENSUITE SU
R NEWLINE."
```

P. — Tu n'oublies pas de mettre une PAUSE entre les deux phrases !

O. — Non, bien sûr !... Il faut quand même qu'on ait le temps de lire la première phrase :

```
95 PAUSE 250
```

P. — Bien !... Maintenant que ceci est réglé, je vais t'expliquer ce qu'on appelle les chaînes de caractères !

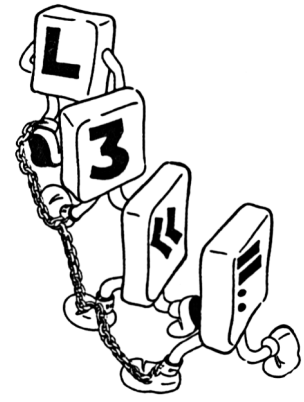
O. — Vas-y, je suis tout ouïe !

P. — Une chaîne de caractères, c'est une suite de caractères que l'on considère comme un tout et qui, une fois mise en mémoire, peut-être rappelée à l'aide d'une simple variable suivie du signe $!

O. — Tu me montres un exemple ?

P. — Tout de suite !... D'ailleurs, on va choisir celui du jeu que nous sommes en train de bâtir !... Dans ce jeu, il faut taper correctement le mot DEPART pour que ce dernier ait lieu. Il faut donc inscrire ce mot en mémoire et ensuite venir comparer, à la fin de chaque frappe, si ce qui a été écrit est identique à ce qui a été mis en mémoire. Cela se réalise de la façon suivante :

```
110 LET A$="DEPART"
```



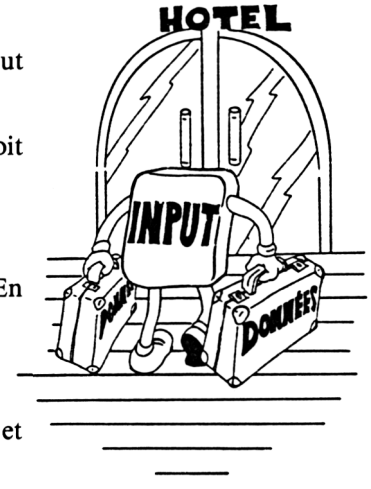
O. — C'est comme quand on affecte une valeur à une variable !... Les deux seules différences ici, c'est que la variable est suivie du symbole $ et qu'au lieu d'un nombre il s'agit de lettres !

P. — En fait, tu n'es pas limité aux seules lettres quand tu crées une chaîne de caractères !... Tu peux y placer également des chiffres, des symboles graphiques, des signes de ponctuation, etc... !

O. — C'est drôlement pratique ça !

P. — Et comment donc !... Pour continuer notre exemple, il suffit maintenant de s'assurer que ce qui va être frappé correspond à la chaîne que nous venons de définir. Tu peux vérifier cela ainsi :

```
120 INPUT B$
130 IF B$=A$ THEN GOTO '50
140 GOTO 120
```



O. — Tiens, une nouvelle instruction que je ne connais pas !... INPUT, ça ne veut pas dire « entrée » en anglais ?

P. — Si fait !... Dans ce cas présent, cette instruction indique que l'on doit « entrer » des données à l'aide du clavier !

O. — Et c'est quoi, des données ?

P. — Ce peut être un nombre, un symbole, un mot bref, n'importe quoi !... En l'occurence, il s'agit du mot DEPART !

O. — Et à quoi correspond B$?... A une seconde chaîne de caractères ?

P. — Exactement !... Ainsi, tu n'as plus qu'à comparer entre elles les chaînes A$ et B$ pour voir si elles sont identiques !

O. — Dans le fond, c'est tout simple !

P. — Mais oui !... Maintenant, as-tu remarqué les guillemets qui encadrent le curseur ?... Sais-tu à quoi ils servent ?

O. — Non !

P. — Ils indiquent que ce que tu vas taper au clavier va constituer une chaîne de caractères !

O. — Ah bon !... Ça a une importance de le savoir ?

P. — Oh que oui !... Lorsque apparaissent les guillemets autour du curseur, il ne t'est plus possible de taper autre chose que ce que l'on attend de toi !

O. — C'est pas drôle !

P. — Je rajouterais même qu'il n'est plus possible d'interrompre le déroulement du programme en appuyant sur BREAK.

O. — Et qu'est-ce qui se passerait si je me trompais ?

P. — Essaye donc pour voir !

O. — Ça y est, je viens de me tromper !... J'ai oublié une lettre du mot DEPART !

P. — Et alors ?

O. — Il ne se passe rien !... C'est normal ?

P. — Absolument !... Tant que tu n'auras pas écrit correctement le mot de passe, tu ne pourras pas sortir de la boucle 12Ø, 14Ø !

O. — C'est super ce « truc », heu... pardon, cette chaîne de caractères !... Ça me fait penser à la phrase « SESAME OUVRE-TOI », de l'histoire d'Ali Baba et les quarante voleurs, qu'il fallait prononcer devant la porte de la caverne avant de pouvoir y entrer !

P. — Je suis content que cela t'ouvre des horizons nouveaux !... Parlant d'ouvrir, ne crois-tu pas qu'il serait temps de libérer les chevaux de leurs boîtes de départ !

O. — Tu as raison, ils doivent s'impatienter les pauvres !... Voici donc de quoi leur donner le départ :

```
 15Ø FOR L=Ø TO 18
 16Ø PRINT AT L,5;"  "
 17Ø NEXT L
 18Ø PRINT
 19Ø PRINT "LES CHEVAUX SONT PAR
TIS."
```

P. — Bravo pour la suppression de la ligne de départ !... Par contre, pour le texte, ce n'est pas encore tout à fait au point !

O. — Comme tu dis !... Il faut que j'efface tout ce qui est inutile sur les lignes 2Ø et 21 !

P. — Avant que tu ne modifies le petit programme que tu viens d'écrire, je voudrais te montrer quelque chose d'intéressant !

O. — De quoi s'agit-il ?

P. — De l'influence de la virgule sur l'instruction PRINT !

O. — ? ? ?

P. — Je m'explique !... Si, après une instruction PRINT, tu places une virgule, l'inscription sur l'écran se fera automatiquement au milieu de la ligne et non au début !

O. — Quelle idée !

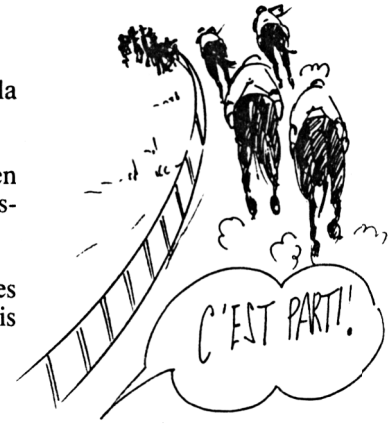
P. — Attends, tu vas comprendre !... Si maintenant, tu places deux virgules à la suite d'une instruction PRINT, l'inscription se fait automatiquement au début de la ligne suivante !

O. — Tu veux dire qu'une virgule décale ce qui doit être imprimé d'une demi-ligne ?

P. — Exactement !... Et il n'y a pas de limites !... Tu peux en placer autant que tu le souhaites. C'est à toi de juger si la solution des virgules est plus économique qu'une autre solution !

O. — Intéressant !... Je peux donc remplacer les deux lignes 18Ø et 19Ø par une seule :

```
180 PRINT ,,"LES CHEVAUX SONT P
ARTIS."
```



P. — Comme tu peux le constater, cela ne change rien aux résultats !... Donc, cela remplace effectivement tes deux lignes précédentes !

O. — Il ne me reste plus qu'à effacer ce qui est inutile !... J'ai envie de le faire en plaçant toute une série d'espaces blancs à la suite de « C'EST PARTI ». Qu'en penses-tu ?

P. — J'en pense que ce n'est pas une méthode très économique que tu me proposes là !... Je vais t'en soumettre une autre qui, je suis sûr, va te plaire beaucoup plus. Ecris donc ceci :

```
180 GOSUB 300
190 PRINT AT 20,0;"LES CHEVAUX
SONT PARTIS."
300 FOR L=20 TO 21
310 FOR C=0 TO 31
320 PRINT AT L,C;" "
330 NEXT C
340 NEXT L
350 RETURN
```

O. — Oh, super !... Ça efface progressivement le texte sur l'écran !

P. — De surcroît, comme il s'agit d'un sous-programme, tu peux y faire appel autant de fois que tu le souhaites !

O. — C'est vrai !... D'ailleurs, ça me donne une idée !... Je vais m'en servir pour effacer progressivement la première phrase à l'aide de la ligne suivante :

```
97 GOSUB 300
```

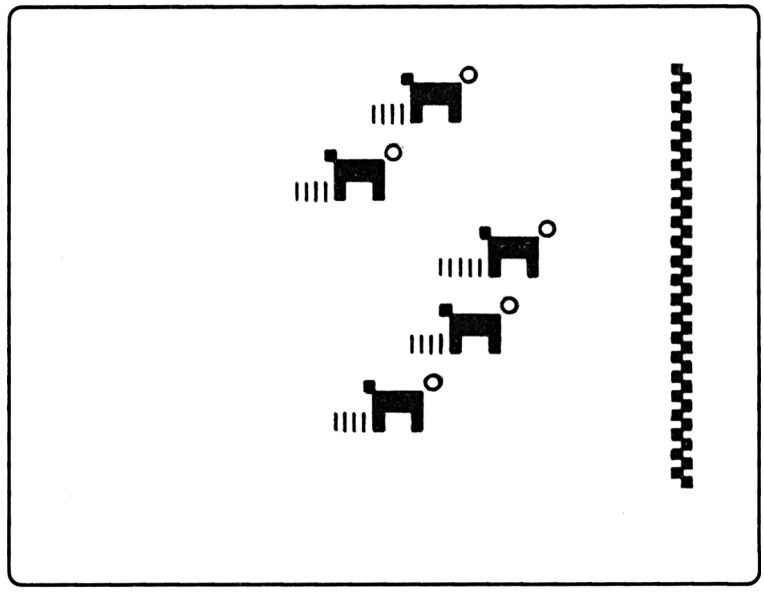
P. — C'est du plus bel effet !... Félicitations !

O. — Bien !... Il ne nous reste plus qu'à faire courir les chevaux et à afficher le gagnant !

O. — Ça, tu me le laisses !... Il me suffit de reprendre le programme des camions dans lequel on employait des variables indicées et de le modifier pour qu'il fasse avancer des chevaux !... Tiens, dis-moi un peu ce que tu penses de cela :

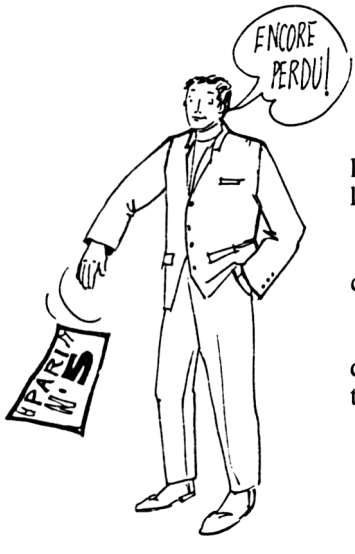
```
200 DIM C(5)
210 LET N=INT (RND*5+1)
220 LET C(N)=C(N)+1
230 PRINT AT 4*N-4,C(N);"▪   O"
240 PRINT AT 4*N-3,C(N);" ▮";CH
R$ (156+N);"▮"
250 PRINT AT 4*N-2,C(N);" ▮ ▮"
260 IF C(N)=27 THEN GOTO 400
270 GOTO 210
```

P. — Très bien !... Il ne manque plus que l'affichage du numéro gagnant et le jeu est terminé !



O. — C'est comme si c'était fait !... Et le gagnant c'est le... :

```
 400 GOSUB 300
 410 PRINT AT 20,0;"C EST LE NUM
ERO ";N;" QUI A REMPORTELA COURS
E."
 420 STOP
```



P. — Le 5 !... J'ai perdu, j'avais misé sur le 3 !

O. — Il est super ce jeu !

Et c'est vrai qu'il n'était pas trop mal le jeu que nous venions d'inventer... surtout si l'on en croit le temps que nous avons ensuite passé à jouer avec !... J'en faisais justement la remarque à Olivier :

P. — Dis donc, tu as vu l'heure !... Il est presque midi !... Il va être temps d'aller déjeûner !

O. — Tiens, tu viens de me donner une idée en parlant de l'heure !... Pourquoi ne dessinerait-on pas un chronomètre ?... On pourrait l'utiliser dans un jeu où il faut effectuer un maximum d'opérations en un temps limité !

P. — Excellente idée !... Reste à trouver le jeu qui conviendrait !

Le chronométrage du temps

O. — Alors là, ne te fais aucun souci !... Que penserais-tu de ballons qu'il faudrait essayer de crever en un temps limité ?

P. — Pourquoi pas ?... Et comment envisagerais-tu de les crever ?

O. — Avec une croix, tiens pardi !

P. — Donc, si je te suis, tu déplacerais cette croix de façon à l'amener sur le ballon que tu veux faire éclater ?

O. — Exactement !

P. — Et donc, il va falloir dessiner un chronomètre pour que tu saches à chaque instant combien de temps il te reste !

O. — C'est bien ça !... Quant aux résultats, c'est-à-dire le nombre de ballons crevés, il suffira de les afficher après que le temps limite se soit écoulé !

P. — Il faudra aussi écrire un programme d'éclatement pour les ballons !... Pas mal de travail en perspective sur ce jeu !

O. — N'est-ce pas ?... Alors, autant commencer tout de suite !... Voici déjà l'enceinte qui va contenir les ballons :

```
10 FOR A=0 TO 21
20 PRINT AT 0,A;"■";AT 21,A;"■
";AT A,0;"■";AT A,21;"■"
30 NEXT A
```

P. — Je vois que tu t'es souvenu de la méthode qui permet de dessiner un carré avec le minimum d'instructions !... C'est bien !

O. — Aux ballons maintenant !... J'ai envie d'en dessiner une centaine !... Qu'est-ce que tu en penses ?

P. — C'est beaucoup trop !... Essaye plutôt une vingtaine ou une trentaine !... Cela devrait suffire !... Sinon, cela risque d'être trop facile s'il y en a trop !

O. — Bon !... J'en dessine donc 25 au hasard :

```
40 FOR N=1 TO 25
50 PRINT AT INT (RND*20+1),INT
(RND*20+1);"O"
60 NEXT N
```

P. — As-tu compté combien il en apparaissait à l'écran ?

O. — Non, pourquoi ?

P. — Compte-les donc !... Cela m'étonnerait fort qu'il y en ait 25 au total !

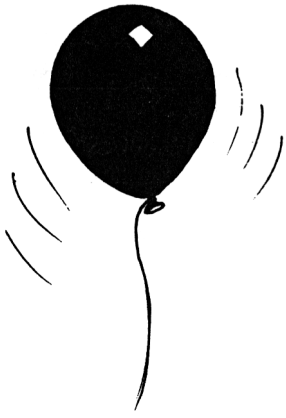
O. — Bon, je compte !... Tu as raison, il n'y en a que 23 !... C'est pas normal, ça !

P. — Et si pourtant !... Cela provient du fait que tu dessines plusieurs ballons les uns sur les autres !

O. — Ah, c'est vrai, il n'y a rien pour empêcher cela !... Tu connais une solution, toi ?

P. — Oui, et je vais te l'enseigner !... Il suffit de vérifier avant de dessiner un ballon sur l'écran s'il n'y en a pas déjà un à cet emplacement !... Cela se fait avec l'instruction PEEK que tu connais bien :

```
 50 LET L=INT (RND*20+1)
 60 LET C=INT (RND*20+1)
 70 IF PEEK (1+C+33*L+256*PEEK
16397+PEEK 16396)=CODE "O" THEN
GOTO 50
 80 PRINT AT L,C;"O"
 90 NEXT N
```



O. — Effectivement, ça marche !... Dans le fond, c'est simple !... Tu vérifies si l'emplacement désigné par L et C ne contient pas déjà un ballon. Et si c'est le cas, tu retournes chercher d'autres valeurs pour L et C !... Je m'en souviendrai !

P. — Maintenant, veux-tu que tes ballons restent immobiles, ou bien aimerais-tu qu'ils bougent ?

O. — Qu'ils bougent, mais pas tous à la fois !... L'un après l'autre, ça serait bien !

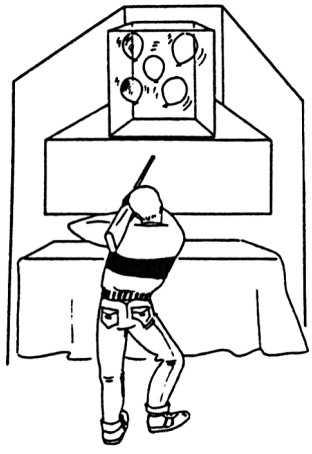
P. — Alors voici un petit programme qui devrait te plaire :

```
100 LET L=INT (RND*20+1)
110 LET C=INT (RND*20+1)
120 IF PEEK (1+C+33*L+256*PEEK
16397+PEEK 16396)=CODE " " THEN
GOTO 100
130 PRINT AT L,C;" "
140 LET L=INT (RND*20+1)
150 LET C=INT (RND*20+1)
160 IF PEEK (1+C+33*L+256*PEEK
16397+PEEK 16396)=CODE "O" THEN
GOTO 130
170 PRINT AT L,C;"O"
```

P. — N'aurais-tu rien oublié, par hasard ?

O. — Oh si !... De fermer la boucle :

```
180 GOTO 100
```



O. — Super !... Les ballons disparaissent et réapparaissent à tour de rôle !... Ça ne va pas être facile de les crever maintenant !

P. — Le jeu n'en sera que plus drôle, ne crois-tu pas ?

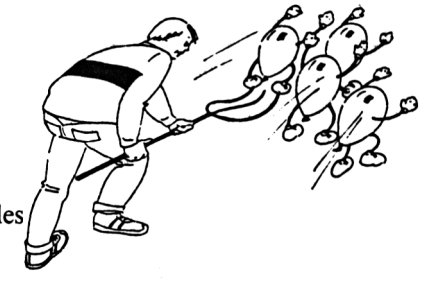
O. — Si, bien sûr !... Je vais d'ailleurs de ce pas écrire le programme de déplacement de la croix :

```
180 PRINT AT Y,X;" "
190 IF INKEY$="5" AND X>1 THEN
LET X=X-1
```

```
200 IF INKEY$="6" AND Y<20 THEN
LET Y=Y+1
210 IF INKEY$="7" AND Y>1 THEN
LET Y=Y-1
220 IF INKEY$="8" AND X<20 THEN
LET X=X+1
230 PRINT AT Y,X;"+"
240 GOTO 100
```

O. — ... je sais !... J'ai encore oublié de définir les variables X et Y !... Voilà, c'est fait :

```
1 LET X=10
2 LET Y=10
```



P. — Et alors ?

O. — Ben, ce n'est pas génial !... Le déplacement du signe + qui me sert à crever les ballons n'avance pas assez vite !... Je suppose que c'est dû aux lignes 120 et 160 ?

P. — Exact !... Toutefois, il y a une solution pour arranger cela !... C'est d'ailleurs la même que celle dont nous nous sommes servi dans la course de voitures !

O. — Tu veux parler de celle qui consiste à placer les instructions de déplacement de la croix dans un sous-programme ?

P. — C'est ça !... En disposant judicieusement les deux lignes d'appel à ce sous-programme, c'est-à-dire en en plaçant une dans chaque boucle de recherche 100-120 et 140-160, tu devrais obtenir une vitesse de déplacement satisfaisante !

O. — Ok, j'essaye tout de suite !... Voici pour commencer les deux lignes d'appel du sous-programme :

```
115 GOSUB 400
155 GOSUB 400
```

O. — ... suivies du sous-programme en question qui vient remplacer les lignes 180 à 230 :

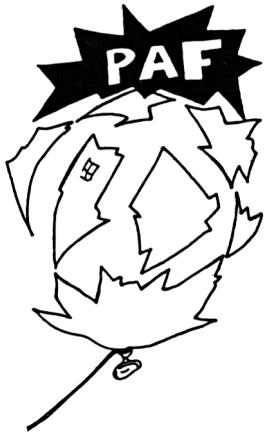
```
400 PRINT AT Y,X;" "
410 IF INKEY$="5" AND X>1 THEN
LET X=X-1
420 IF INKEY$="6" AND Y<20 THEN
LET Y=Y+1
430 IF INKEY$="7" AND Y>1 THEN
LET Y=Y-1
440 IF INKEY$="8" AND X<20 THEN
LET X=X+1
450 PRINT AT Y,X;"+"
460 RETURN
```

O. — Là maintenant, c'est bon !... La vitesse est tout à fait satisfaisante !

P. — Alors, si ça te convient, on passe au programme d'éclatement !... Veux-tu que je t'aide pour cela ?

O. — Non merci, ça va aller !... Je sais qu'il faut placer la ligne de détection de l'impact avant celle de l'impression de la croix !... Voici ce que ça donne :

```
445 IF PEEK (1+X+33*Y+256*PEEK
16397+PEEK 16396)=CODE "O" THEN
GOSUB 500
500 PRINT AT Y,X-1;"(▒)"
510 PAUSE 10
520 PRINT AT Y,X-1;"PAF";AT Y-1
,X-1;"■ ■";AT Y+1,X-1;"■ ■"
530 PAUSE 10
540 PRINT AT Y,X-1;"    ";AT Y-1
,X-1;"■  ■";AT Y+1,X-1;"■  ■"
550 PAUSE 10
560 PRINT AT Y-1,X-1;"    ";AT Y
+1,X-1;"    "
570 RETURN
```



O. — ... c'est hyper génial !... Comment trouves-tu mon idée d'écrire « PAF » au moment où le ballon explose ?

P. — Remarquable !... C'est ce que je pourrais appeler un bruitage visuel... Par contre, je viens de remarquer un défaut dans notre jeu !

O. — Moi aussi !... Quand il y a deux ballons placés côte à côte, l'éclatement de l'un entraîne l'effacement de son voisin !... C'est trop bête de perdre ainsi un ballon !

P. — N'est-ce pas ?... Cela te fera un point de moins si je comprends bien ?

O. — Ben oui !

P. — De surcroît, il se pose un problème du même genre lorsque le ballon est placé en bordure de l'enceinte et qu'il éclate. Une partie de la paroi disparaît !

O. — C'est gênant !

P. — Pour supprimer ces deux défauts, je te propose donc d'imprimer les ballons de deux cases en deux cases, et de laisser un espace entre le cadre et les ballons qui s'en rapprochent le plus !

O. — Adjugé !... Par contre, il va falloir réécrire tout le début du programme !... Ça, c'est moins drôle !... Enfin, allons-y en commençant par le cadre :

```
10 FOR A=0 TO 20
20 PRINT AT 0,A;"■";AT 20,A;"■
";AT A,0;"■";AT A,20,"■"
```

O. — ... puis en continuant par le programme d'apparition et de disparition des ballons :

```
 50 LET L=2*INT (RND*9+1)
 60 LET C=2*INT (RND*9+1)
100 LET L=2*INT (RND*9+1)
110 LET C=2*INT (RND*9+1)
140 LET L=2*INT (RND*9+1)
150 LET C=2*INT (RND*9+1)
```

O. — ... et en terminant avec le sous-programme de déplacement dans lequel il faut que je modifie deux butées sur quatre :

```
420 IF INKEY$="6" AND Y<19 THEN
LET Y=Y+1
440 IF INKEY$="8" AND X<19 THEN
LET X=X+1
```

P. — Et voilà !... Maintenant, on est fin prêt pour aborder l'étude du chronomètre !

O. — Enfin presque !... Je viens de constater deux défauts !... Le premier, tout simple, se produit lorsque l'on dessine la croix sur l'écran. Si un ballon se trouve en dessous, il est aussitôt effacé !... Pour remédier à cela, je t'ai décalé d'une unité les deux valeurs initiales de X et Y :

```
  1 LET X=9
  2 LET Y=9
```

P. — Voilà pour le premier défaut !... Et le second ?

O. — Là, il s'agit de ballons supplémentaires qui apparaissent à la fin du jeu et qui faussent tout !

P. — Attends que je réfléchisse à cela !... Ok, je viens de trouver !... En fait, ce défaut existe tout le temps mais il n'est surtout visible que quand il ne reste que quelques ballons. Il a pour origine la ligne 120 !... Si tu analyses bien cette ligne, tu vas constater qu'avant d'effacer un ballon on recherche les espaces, mais qu'on oublie de rechercher la croix !

O. — Je viens de comprendre !... Si donc on efface la croix, ce qui ne dure qu'une fraction de seconde, on autorise l'impression d'un ballon. Et comme il n'y en a pas eu d'effacé, c'est un ballon supplémentaire qui vient d'être créé !... Je t'arrange ça tout de suite :

```
 120 LET J=PEEK (1+C+33*L+256*PE
EK 16397+PEEK 16396)
 125 IF J=CODE " " OR J=CODE "+"
 THEN GOTO 100
 160 LET K=PEEK (1+C+33*L+256*PE
EK 16397+PEEK 16396)
 165 IF K=CODE "O" OR K=CODE "+"
 THEN GOTO 140
```

P. — Bien !... Maintenant, on passe au chronomètre pour de bon !... On va d'abord commencer par dessiner son boîtier !... Voici ce qui devrait faire l'affaire :

```
  92 PRINT AT 2,24;"▬▬▬▬▬▬";AT 3
,24;"▌ :  ▌";AT 4,24;"▬▬▬▬▬▬"
  94 PRINT AT Ø,24;"TEMPS";AT 1,
24;"ALLOUE"
```

O. — Super !... Maintenant, il ne reste plus qu'à y mettre les chiffres !

P. — Comme tu dis !... L'ennui, c'est que c'est plus facile à dire qu'à faire !... Je n'ai trouvé que la méthode suivante qui, je dois l'avouer, n'est pas géniale !... J'espère que tu en trouveras une autre tout seul !

O. — Dis toujours !

P. — D'abord, il faut définir deux variables, une pour les minutes et une pour les secondes, variables que je vais placer dans un nouveau sous-programme qu'on va baptiser 6ØØ :

```
 6ØØ LET M=INT (T/6Ø)
 61Ø LET S=T-6Ø*M
```

O. — Ok, ça je comprends !... Je suppose que T est le nombre de secondes dont on dispose au début du jeu ?

P. — C'est cela !... D'ailleurs, il ne faut pas que j'oublie de définir cette variable :

```
   3 LET T=9Ø
```

P. — Bien, voilà qui est fait !... Maintenant, il ne reste plus qu'à afficher la valeur des deux variables M et S à l'endroit prévu. Voici déjà pour les minutes :

```
 62Ø PRINT AT 3,25;M
```

P. — ... et voici pour les secondes :

```
 63Ø IF S>=1Ø THEN PRINT AT 3,27
;S
 64Ø IF S<1Ø THEN PRINT AT 3,27;
"Ø";S
```

O. — Tiens, tu as besoin de deux lignes pour faire ça ?

P. — Eh oui !... Tu remarqueras que la seconde est là pour décaler d'un espace vers la droite le chiffre des secondes, lorsqu'il est compris entre Ø et 9 !

O. — Ah, c'est pour ça !

P. — Bien !... Restent encore deux lignes à écrire pour terminer ce sous-programme. La première va nous servir à diminuer T d'une unité à chaque parcours de ce sous-programme. Quant à la seconde, il s'agit de la ligne qui va nous renvoyer sur le sous-programme d'affichage des résultats !... Les voici toutes les deux :

```
650 LET T=T-1
660 IF T=-1 THEN GOTO 700
670 RETURN
```

O. — Il reste encore à placer la ligne d'instruction qui va faire appel à ce sous-programme 600. J'avoue que je ne vois pas très bien où ?

P. — Moi si !... Il faut obligatoirement qu'elle soit quelque part dans le sous-programme 400 qui est, je te le rappelle, le seul qui soit parcouru systématiquement !

O. — Dans celui qui commande le déplacement de la croix ?

P. — C'est bien cela !... Je te suggère la ligne 455, mais tout autre emplacement est aussi valable :

```
455 GOSUB 600
```

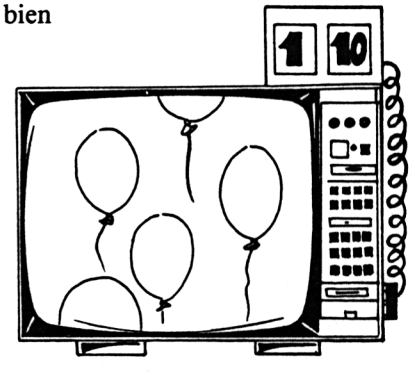
O. — Ça y est, ça marche !... Mais alors, qu'est-ce que ça ralentit le déplacement de la croix !... On ne peut rien faire pour améliorer ça ?

P. — Si, sûrement !... Mais dans le moment présent, je ne vois pas comment !

O. — Ok !... Je chercherai une solution plus tard !... Par contre, j'aimerais bien écrire les lignes d'affichage du nombre de ballons crevés !

P. — Très bonne idée !... Fais-le comme tu l'entends !

O. — Merci !... Voici donc le sous-programme 700 :




```
700 PRINT AT 10,23;"VOUS AVEZ";
AT 11,23;"CREVE ";B;AT 12,23;"BA
LLONS."
710 STOP
```

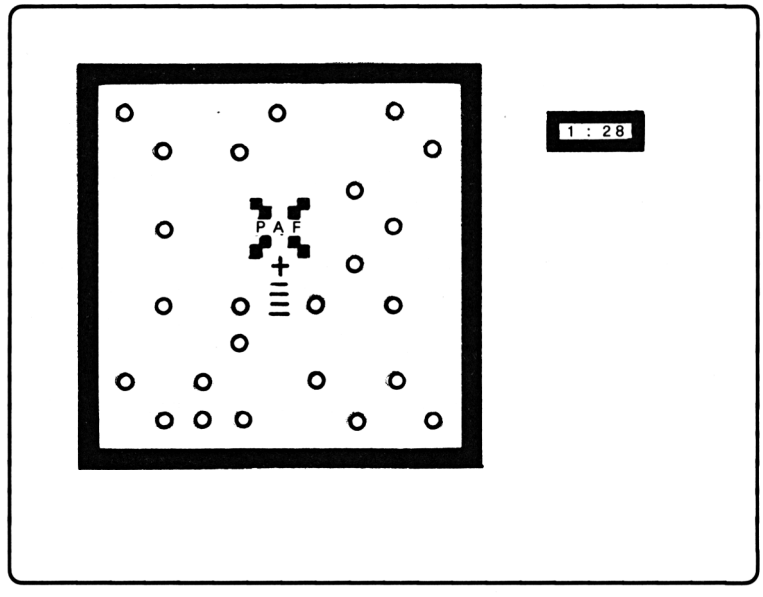
O. — ... suivi de la ligne de définition de la variable B qui correspond aux nombres de ballons crevés :

```
4 LET B=0
```

O. — ... suivie à son tour de celle qui effectue la comptabilité de tout cela. Tu remarqueras que je l'ai placée à la fin du sous-programme d'éclatement :

```
565 LET B=B+1
```

P. — C'est parfait !... J'espère que tu es content de ce dernier jeu !



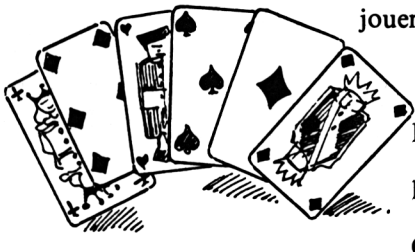
O. — Et comment !... Je viens de faire un score de 24 ballons !... Crois-tu que tu peux faire mieux ?

P. — Je ne sais pas !... Je vais essayer en tout cas !

Et c'est ainsi que pendant toute une heure, j'essayai vainement de battre ce record si facilement acquis par Olivier. Mon score s'établissait invariablement autour de 21 ou 22 ballons !... Las, je lui lançai :

P. — Tout cela, c'est bien joli, mais depuis une heure nous ne progressons plus !... Quelle est la suite du programme ?

O. — J'aimerais que tu me montres comment on fait pour dessiner des cartes à jouer !



Les cartes à jouer

P. — Ok !... Ce que tu me demandes là est relativement facile à faire !

O. — Attends, je ne t'ai pas tout dit !... Je voudrais en profiter pour faire un jeu de poker, c'est-à-dire afficher cinq cartes à l'écran et pouvoir en remplacer jusqu'à trois quand la main n'est pas bonne !

P. — Ouille !... Là, c'est nettement moins facile !... Il va falloir bâtir un véritable jeu comprenant entre 100 et 200 lignes d'instruction !... Cela ne te fait pas peur ?

O. — Non, bien au contraire !

P. — Alors, puisqu'il en est ainsi, on y va !... Pour la construction de ce jeu, on va employer toute une série de sous-programmes. A vue de nez, j'en vois au moins sept ou huit : un pour le tirage au sort des cartes, un pour dessiner le pourtour de chaque carte, un pour leur valeur, un pour leur couleur, un pour l'effacement du texte, un autre pour

l'effacement des cartes, un pour la remise en jeu, et peut-être d'autres auxquels je ne pense pas dans l'immédiat !

O. — Tout ça, c'est très chouette !... J'ai même envie de commencer !... Tu veux bien ?

P. — Mais comment donc !... Après vous, très cher !

O. — Je vais m'amuser un peu à faire dialoguer la machine !... Que penses-tu de ça ?

```
10 PRINT "CECI EST UN JEU DE P
OKER "
20 PAUSE 100
30 FOR C=0 TO 31
40 PRINT AT 0,C;" "
50 NEXT C
60 PRINT AT 0,0;"VOICI CINQ CA
RTES."
```

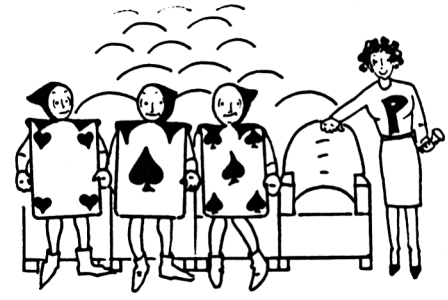
P. — Bravo !... C'est un bon début !... Toutefois, je te conseille de placer le programme d'effacement du texte dans un sous-programme, car on risque d'y faire appel souvent !

O. — C'est vrai !... Inutile de gaspiller la place en mémoire !... A ton avis, quel numéro dois-je donner à ce sous-programme ?

P. — Au moins 300 ou 400, étant donné le grand nombre d'opérations qu'il y a lieu de programmer avant !

O. — Ok !... Je vais être prudent et choisir 400 !... Comme ça, on disposera d'un peu plus de place pour le programme principal !... Voici donc la même chose que précédemment, mais cette fois-ci avec un sous-programme :

```
30 GOSUB 400
40 PRINT AT 0,0;"VOICI CINQ CA
RTES "
400 FOR C=0 TO 31
410 PRINT AT 0,C;" "
420 NEXT C
430 RETURN
```



P. — Très bien !... Comme tu l'annonces si bien, il ne te reste plus qu'à dessiner tes cinq cartes !

O. — C'est moins facile à programmer qu'à dire !... Pour commencer, je voudrais placer les cinq cartes côte à côte au milieu de la hauteur de l'écran !

P. — Cela te fait une moyenne de 5 carreaux par carte. Si l'on tient compte du cadre noir que l'on va tracer tout autour, il ne va te rester que 4 carreaux.

O. — Je vais donc dessiner des cartes de 4 carreaux de largeur par 6 carreaux de hauteur, bordés d'un demi-carreau noir !... Ça devrait aller !

P. — Bien !... Maintenant, je vais te conseiller d'employer une variable P pour positionner chacune de tes cartes dans l'écran. Tu pourrais par exemple affecter à cette variable la valeur de la colonne de gauche de chaque carte, ce qui, dans ton cas, correspondrait aux colonnes 2, 8, 14, 20 et 26.

O. — Bonne idée !... Et ainsi, il me suffit d'écrire pour tracer le cadre de chaque carte :

```
50 FOR P=2 TO 26 STEP 6
60 GOSUB 500
70 NEXT P
```

O. — ... en faisant appel bien sûr à un sous-programme !... Tu vois que j'ai bien appris ma leçon !

```
500 LET I=2*P-1
510 LET J=15
520 FOR X=0 TO 9
530 PLOT I+X,J
540 PLOT I+X,J+13
550 NEXT X
560 FOR Y=0 TO 13
570 PLOT I,J+Y
580 PLOT I+9,J+Y
590 NEXT Y
595 RETURN
```

P. — C'est ce qu'il faut, si tu veux progresser !... Je note au passage qu'à la ligne 500, tu as pensé à effectuer la conversion entre les coordonnées horizontales de l'instruction PRINT avec celles de l'instruction PLOT !

O. — Eh oui !... Par contre, je ne me suis pas occupé de convertir les coordonnées verticales, vu qu'on dessine le cadre toujours sur les mêmes demi-lignes 15 et 28 !

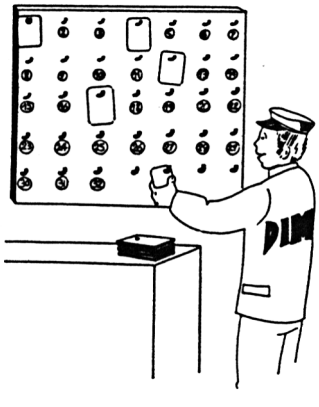
P. — Bien !... Passons maintenant au sous-programme de tirage au sort de la valeur des cartes !

O. — Comment on peut faire un « truc » pareil ?... Ça ne va pas être facile de tirer 32 cartes au hasard sans jamais obtenir deux fois la même !

P. — Le « truc », comme tu dis, consiste à attribuer à chaque carte un nombre, compris entre 1 et 32 et à vérifier dans un tableau si ce nombre n'a pas été déjà tiré !

O. — Hou-là-là, quelle complication !

P. — Mais non !... Il va nous suffire de faire appel aux variables indicées que tu connais bien pour les avoir déjà employées deux ou trois fois déjà dans d'autres jeux !



O. — Aïe, aïe, aïe !... Je ne les aime pas celles-là !

P. — Et pourtant, elles sont si pratiques !... Tout d'abord je te rappelle qu'on doit dimensionner un tableau avec l'instruction DIM !... Pour ce jeu, je te conseille de placer cette instruction juste avant la ligne 40, car c'est en ce point-ci du programme que se fera la remise en jeu automatique. Tu n'as pas oublié que l'instruction DIM plaçait toutes les variables du tableau à zéro, en même temps qu'elle le dimensionnait, n'est-ce pas ?

O. — Non, non !

P. — Ensuite, il suffit de tirer une valeur au hasard entre 1 et 32 et d'aller vérifier dans le tableau des 32 variables si la variable correspondante est bien égale à 0, ce qui indique qu'elle n'a pas encore été tirée. Cette recherche étant faite, il faut alors affecter une valeur quelconque à cette variable du tableau pour bien indiquer que la carte en question a déjà été tirée !

O. — C'est tout ?... Ce n'est pas plus compliqué que cela ?

P. — Et non !... D'ailleurs, voici la ligne de dimensionnement du tableau :

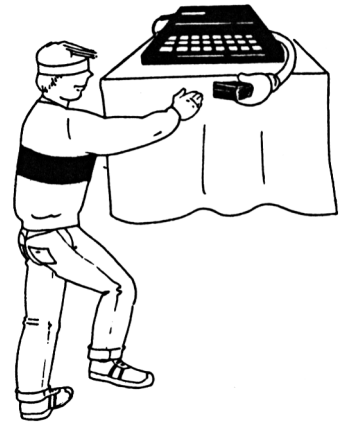
```
35 DIM V(32)
```

P. — ... suivie du sous-programme de tirage au sort des 32 cartes :

```
600 LET N=INT (RND*32+1)
610 IF V(N)=1 THEN GOTO 600
620 LET V(N)=1
630 RETURN
```

P. — ... suivi à son tour par la ligne d'appel de ce sous-programme !... Tu noteras qu'il a fallu que je renumérote la ligne d'instructions NEXT P :

```
70 GOSUB 600
80 NEXT P
```



O. — Je n'aurais pas cru que ça puisse être si simple !

P. — Reconnais qu'elles sont bien pratiques ces variables indicées !

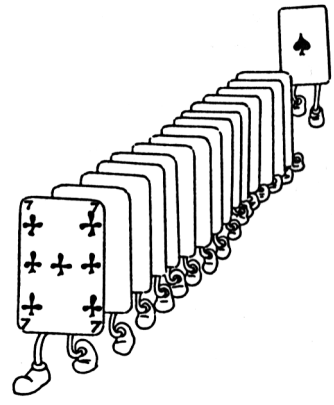
O. — C'est vrai !

P. — Maintenant, il va falloir que je te montre comment on peut affecter une valeur et une couleur à une carte en fonction du numéro que l'on a tiré au hasard !

O. — J'imagine que, pour ça, tu fais appel à une sorte de tableau de correspondance ?

P. — C'est exactement cela !... Je vais t'indiquer celui que j'ai choisi, mais rien bien entendu ne t'empêche d'en établir un autre.

VALEUR	COULEUR			
	Trèfle	Carreau	Cœur	Pique
7	1	9	17	25
8	2	10	18	26
9	3	11	19	27
10	4	12	20	28
V	5	13	21	29
D	6	14	22	30
R	7	15	23	31
AS	8	16	24	32



P. — Comme tu peux le remarquer, le chiffre le plus faible est affecté au 7 de trèfle, et le plus fort à l'as de pique !

O. — Je vois bien !... Par contre, moi, j'aurais affecté les chiffres 1, 2, 3 et 4 respec-

tivement aux sept de trèfle, de carreau, de cœur et de pique, et non comme tu l'as fait aux sept, huit, neuf et dix de trèfle !

P. — C'est pour cela que je t'ai bien précisé qu'il existait des tas d'autres solutions !... En attendant, je vais te bâtir les deux sous-programmes d'affichage des valeurs et des couleurs des cartes à partir du tableau que je t'ai indiqué. Voici déjà pour le premier sous-programme :

```
 700 FOR W=0 TO 3
 710 IF N=W*8+1 THEN PRINT AT 8,
P;"7";AT 13,P+3;"7"
 720 IF N=W*8+2 THEN PRINT AT 8,
P;"8";AT 13,P+3;"8"
 730 IF N=W*8+3 THEN PRINT AT 8,
P;"9";AT 13,P+3;"9"
 740 IF N=W*8+4 THEN PRINT AT 8,
P;"10";AT 13,P+2;"10"
 750 IF N=W*8+5 THEN PRINT AT 8,
P;"V";AT 13,P+3;"V"
 760 IF N=W*8+6 THEN PRINT AT 8,
P;"D";AT 13,P+3;"D"
 770 IF N=W*8+7 THEN PRINT AT 8,
P;"R";AT 13,P+3;"R"
 780 IF N=W*8+8 THEN PRINT AT 8,
P;"AS";AT 13,P+2;"AS"
 790 NEXT W
 795 RETURN
```

O. — Dis donc, on doit pouvoir faire plus simple !

P. — Sûrement !... Cherche et tu trouveras une bien meilleure solution, j'en suis sûr !

O. — C'est effectivement ce que j'essayerai de faire une fois ce jeu terminé !

P. — Bon !... Maintenant, si on veut voir comment fonctionne le sous-programme qu'on vient d'écrire, il ne faut pas oublier de placer sa ligne d'appel dans la boucle FOR...TO de la ligne 50. Au passage, tu remarqueras que j'en ai profité pour placer également la ligne d'appel du sous-programme d'affichage des couleurs qui va débuter à la ligne 800 :

```
  80 GOSUB 700
  90 GOSUB 800
 100 NEXT P
```

O. — Bien !... Allons-y donc pour ce second sous-programme !

P. — C'est parti !... Voici les quatre couleurs que sont le trèfle, le carreau, le cœur et le pique. Comme tu l'imagines aisément, il n'est pas possible de représenter exactement la forme de chacune d'entre elles à l'aide des symboles graphiques dont on dispose. J'ai, encore une fois, fait du mieux que j'ai pu avec les moyens du bord !... J'avoue d'ailleurs ne pas être trop mécontent du résultat obtenu !

O. — Tu fais voir ?

P. — Tout de suite !... Le temps d'écrire quelques lignes :

```
800 IF N>=1 AND N<=8 THEN PRINT
 AT 9,P;"  ";AT 10,P;"  ";AT
11,P;"  ";AT 12,P;"  "
 810 IF N>=9 AND N<=16 THEN PRIN
T AT 9,P;"  ";AT 10,P;"  ";AT
 11,P;"  ";AT 12,P;"  "
 820 IF N>=17 AND N<=24 THEN PRI
NT AT 9,P;"  ";AT 10,P;"  ";
AT 11,P;"  ";AT 12,P;"  "
 830 IF N>=25 AND N<=32 THEN PRI
NT AT 9,P;"  ";AT 10,P;"  ";A
T 11,P;"  ";AT 12,P;"  "
 840 RETURN
```

P. — ... et voilà !

O. — Ah, c'est pas mal du tout !... C'est super de voir ainsi cinq cartes se dessiner toutes seules à l'écran !

P. — N'est-ce pas ?... Toutefois, nous n'en avons pas terminé avec ce jeu, loin s'en faut !... Il nous faut maintenant aborder une partie assez délicate, j'ai nommé le remplacement des cartes qui ne conviennent pas, et ce jusqu'à concurrence de trois, par d'autres tirées au hasard, bien entendu !

O. — Dans ce cas, il faut numéroter les cartes sur l'écran, sinon comment pourrait-on distinguer celles que l'on veut remplacer de celles que l'on veut conserver !

P. — Très juste !... Voici ce qui devrait faire l'affaire :

```
 55 PRINT AT 16,P;"(";(P+4)/6;"
)"
```

O. — Effectivement, ça marche !... Maintenant, pour ce qui est des cartes que l'on choisit de remplacer par d'autres, je ne vois vraiment pas comment on peut faire !

P. — Ne te fais donc pas de soucis et suis mon raisonnement !... Tout d'abord, il va falloir indiquer à la machine combien de cartes on veut échanger. Ce n'est qu'une fois qu'elle connaîtra le nombre qu'on pourra lui préciser lesquelles il faut éliminer !

O. — Je te suis !... Je crois même pouvoir écrire la première partie de ce programme !... Il me semble que quelque chose comme ça devrait marcher :

```
 110 GOSUB 400
 120 PRINT AT 0,0;"COMBIEN DE CA
RTES VOULEZ-VOUS ?"
 130 PAUSE 100
 140 GOSUB 400
 150 PRINT AT 0,0;"0 , 1 , 2 OU
3 ?"
```

P. — Oui !... Il ne te reste plus qu'à rajouter les lignes d'entrée du nombre de cartes et celles de son traitement !

O. — De quel traitement parles-tu ?

P. — En disant cela, je pense au fait qu'il faut tout d'abord vérifier si le nombre indiqué est bien compris entre Ø et 3, et qu'ensuite il faut agir différemment selon qu'il s'agit d'un Ø ou d'un chiffre compris entre 1 et 3 !

O. — C'est vrai que si l'on frappe un Ø, c'est que l'on est satisfait de son jeu !... Il n'y a donc pas lieu dans ce cas d'aller plus loin !

P. — Exact !... Et voici comment je vois la chose :

```
160 INPUT K
170 IF K=Ø THEN GOTO 1ØØØ
180 IF K>3 THEN GOTO 16Ø
```

O. — Et que trouve-t-on en 1ØØØ ?

P. — Une sorte de sous-programme de remise en jeu qu'on étudiera plus tard !

O. — Ah bon !

P. — Ceci étant fait, on va pouvoir passer à la seconde partie de ce programme, c'est-à-dire celle qui va nous permettre de déterminer les cartes qu'il faut effacer. J'imagine que tu préfères que ce soit moi qui programme cela ?

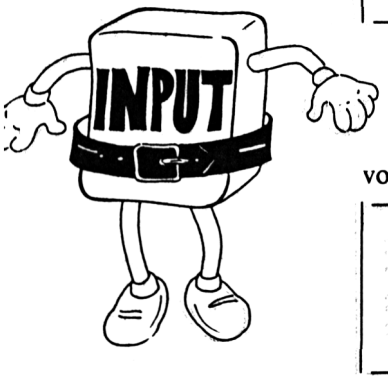
O. — Oh que oui !

P. — On va donc commencer par demander au joueur de préciser le numéro des cartes qu'il veut supprimer :

```
190 GOSUB 4ØØ
2ØØ PRINT AT Ø,Ø;"NUMERO DES CA
RTES A SUPPRIMER:"
```

P. — ... que l'on va faire suivre par une ligne de dimensionnement des variables qui vont correspondre aux cartes à éliminer :

```
21Ø DIM S(K)
```



O. — Jusque-là, ça va !

P. — Ensuite, on va placer une instruction INPUT dans une boucle, de façon à pouvoir frapper plusieurs numéros les uns à la suite des autres :

```
22Ø FOR Z=1 TO K
23Ø INPUT S(Z)
24Ø PRINT AT Z,Ø;"(";S(Z);")"
25Ø NEXT Z
```

O. — Oh, très astucieuse cette façon de faire !... Faudra que je m'en souvienne à l'avenir !

P. — Comme le reste d'ailleurs...

O. — Oui, bien sûr !... En attendant, je vais pouvoir terminer le jeu !... Le seul point qui me gênait c'était la façon d'entrer le numéro des cartes à éliminer !... Pour la suite, je vois les choses ainsi :

```
260 FOR Z=1 TO K
270 LET P=6*S(Z)-4
280 GOSUB 900
290 NEXT Z
```

P. — Je suppose que 900 correspond au sous-programme d'effacement des cartes à éliminer ?

O. — Exact !... D'ailleurs, le voici :

```
900 FOR L=7 TO 14
910 FOR C=P-1 TO P+4
920 PRINT AT L,C;"  "
930 NEXT C
940 NEXT L
950 RETURN
```

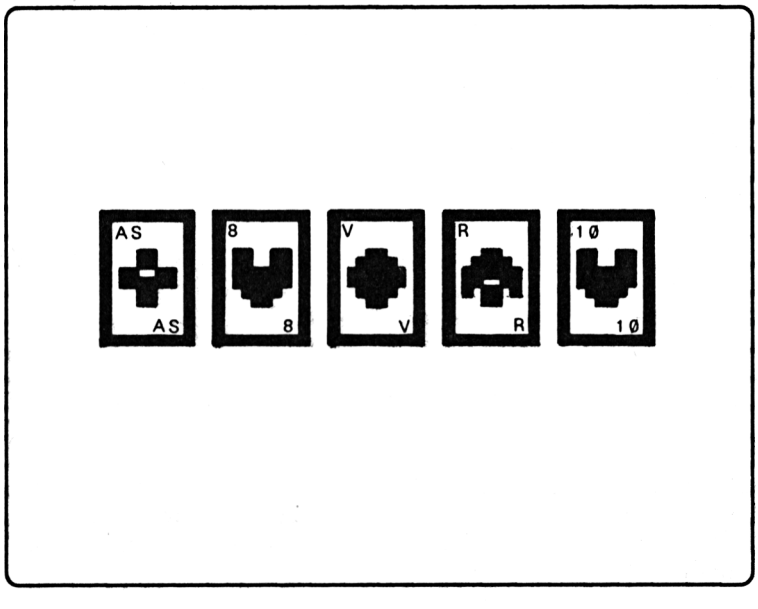
P. — Parfait !... Continue ainsi !

O. — Après l'effacement des cartes, il ne reste plus qu'à en redessiner des nouvelles !... Il faut donc écrire pour cela :

```
300 FOR Z=1 TO K
310 LET P=6*S(Z)-4
320 GOSUB 500
330 GOSUB 600
340 GOSUB 700
350 GOSUB 800
360 NEXT Z
```

P. — Et voilà !... As-tu gagné ?

O. — Non !... Mais je vais recommencer !... Je l'aurai, mon carré d'as !



P. — Ce qu'il te faut maintenant, c'est un programme de remise en jeu !... Je te propose la chose suivante :

```
 370 GOTO 1000
1000 GOSUB 400
1010 PRINT AT 0,0;"UNE AUTRE PAR
TIE ?"
1020 INPUT A$
1030 IF A$="OUI" THEN GOTO 1060
1040 IF A$="NON" THEN STOP
1050 GOTO 1020
1060 GOSUB 400
1070 FOR P=2 TO 26 STEP 6
1080 GOSUB 900
1090 NEXT P
1100 GOTO 35
```

O. — Attends que je comprenne !... Tout d'abord, après avoir effacé le texte précédent, tu demandes si on veut faire une autre partie. Ensuite, si la réponse est oui, tu effaces à nouveau le texte, puis les cinq cartes. Après quoi, tu retournes en 35 pour remettre à zéro les variables du tableau !... Il y a toutefois un point qui m'intrigue : à quoi sert la ligne 1050 ?

P. — A vérifier si tu réponds bien OUI ou NON !... Si ce n'est pas le cas, la réponse est refusée grâce à cette ligne.

O. — C'est pratique comme méthode !... Ainsi, pas de surprise !

P. — N'est-ce pas ?... Avant de te laisser jouer avec, je voudrais rectifier un petit défaut que je viens de constater !... On a complètement oublié de prévoir l'effacement des numéros des cartes à éliminer !.. Voici ce qui devrait résoudre ce petit problème :

```
1005 FOR L=1 TO K
1006 PRINT AT L,0;"    "
1007 NEXT L
```

O. — C'est le cas !... Maintenant, ce que j'aimerais, c'est que tu me montres comment on fait pour écrire un programme qui détecte les combinaisons gagnantes !

P. — C'est très simple !... Tu te creuses la tête !... Si jusqu'à présent je t'ai expliqué certaines méthodes de conception des jeux, il ne faut pas oublier que le travail d'invention, c'est toi qui dois le faire par toi-même !

O. — Bon !... Eh bien, je cherche !

Et il chercha longtemps, très longtemps !... Je crois qu'il serait encore en train de chercher si je n'étais pas intervenu.

P. — Alors, tu t'en sors ?

O. — Pas précisément !... Tu sais, c'est horriblement compliqué à calculer toutes ces combinaisons !...

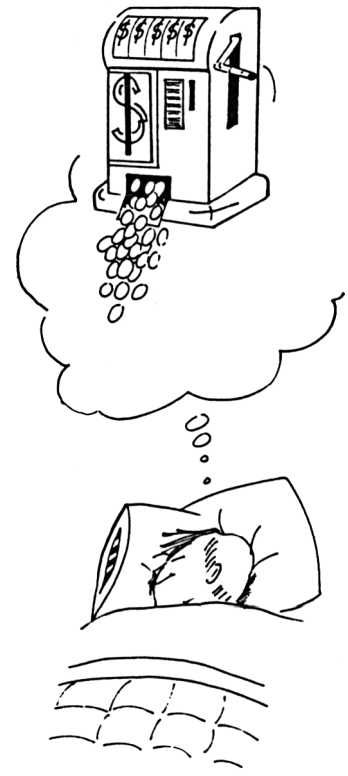
P. — Je ne te le fais pas dire !... En fait, je voulais que tu te rendes compte de la complexité que représente une telle recherche, de façon à être bien conscient que ce n'est pas dix minutes qui peuvent y suffire, ni une heure, ni même une journée. Il faut parfois y consacrer plusieurs semaines avant d'aboutir au résultat recherché !

O. — Alors, ça peut attendre...

P. — C'est bien mon avis !... Et maintenant, que souhaites-tu réaliser !... Je t'accorde encore un jeu et après, c'est fini, tu te débrouilles tout seul !

O. — Si je n'ai plus droit qu'à un jeu, alors, je choisis le « jackpot » !... J'ai cette idée qui me trotte dans la tête depuis plusieurs semaines et j'aimerais bien pouvoir la concrétiser !

P. — Bonne idée !... D'autant plus que cela va nous permettre d'illustrer la méthode de création de gros caractères sur l'écran !



Les gros caractères

O. — C'est vrai que pour faire un « jackpot », il faut de sacrés gros caractères !... Que penserais-tu d'employer le signe dollar ($), la livre anglaise (£) et l'étoile (★) ?

P. — Pourquoi pas si c'est ce qui te plaît !... Et je suppose que tu vas me demander s'il n'existe pas une méthode qui permette d'agrandir les caractères du clavier !

O. — Comment as-tu deviné ?... C'est exactement la question que j'allais te poser !

P. — Je me l'étais déjà posée avant toi !... Oui, il existe un moyen, pas forcément très simple, qui permet d'afficher sur une matrice de 8 × 8 carreaux tous les 256 caractères, symboles ou signes de ponctuation auxquels on a accès à partir du clavier !... Pour cela il faut faire appel à l'instruction PEEK, car on va devoir fouiller un peu dans la mémoire des caractères !

O. — Tu fais voir ?

P. — Tout de suite !... Essaye donc ce petit programme :

```
 10 LET K=7680+8*CODE "£"
 20 FOR L=1 TO 8
 30 LET X=PEEK K
 40 FOR E=7 TO 0 STEP -1
 50 LET Q=INT (X/2**E)
 60 LET X=X-Q*2**E
 70 PRINT AT L,7-E;CHR$ (128*Q)
 80 NEXT E
 90 LET K=K+1
100 NEXT L
```

O. — Hou là là, qu'est-ce qu'il est compliqué !

P. — Je ne te demande pas de le comprendre, mais de l'essayer !

O. — Bon, bon !... Voilà, c'est parti !... Ben, il n'y a rien qui se passe !... Ah si, le signe de la livre anglaise se dessine très lentement !... Trop lentement d'ailleurs !... Il n'y a pas moyen d'accélérer tout ça ?

P. — Non !... C'est le défaut de cette méthode !... On va donc en utiliser une autre plus rapide et plus facile à comprendre !... Toutefois, je voulais que tu connaisses l'existence de celle que je viens de te montrer car elle peut te servir quand tu veux écrire un gros titre sur l'écran !

O. — Comme le nom d'un jeu, par exemple ?

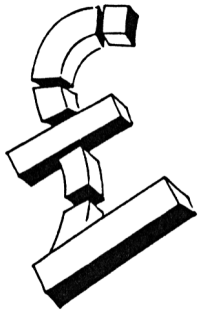
P. — Oui, pourquoi pas !

O. — Bon, alors, tu me montres une autre méthode pour dessiner les caractères !

P. — En fait, tu la connais déjà !... C'est celle qui consiste à employer des chaînes de caractères !

O. — Des chaînes de caractères ?... Là, je ne comprends pas !

P. — Tu vas peut-être comprendre si je te dis qu'il faut associer une chaîne à chaque ligne de dessin du caractère !



O. — Ainsi, pour dessiner un caractère comme la livre anglaise, il me faut... attends, je compte... six lignes, donc six chaînes.

P. — Perdu !... Il n'en faut que cinq, car si tu remarques bien, il y a deux lignes identiques !

O. — Vu !... Maintenant, je comprends à quoi servent les chaînes !

P. — Puisque tu as compris, on pourrait peut-être commencer ce fameux « jack-pot » dont tu me rabats les oreilles !

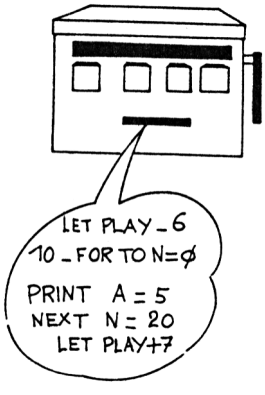
O. — C'est parti !... Voici déjà le corps de l'appareil !... Je te signale que la fente du haut, c'est pour le texte et celle du bas, c'est pour les jetons :

```
10 FOR L=0 TO 21
20 FOR C=0 TO 31
21 IF L=3 AND C=2 THEN LET C=3
0
22 IF L=18 AND C=12 THEN LET C
=20
23 IF L>=7 AND L<=14 THEN GOTO
 25
24 GOTO 30
25 IF C=2 OR C=12 OR C=22 THEN
 LET C=C+8
30 PRINT AT L,C;"■"
40 NEXT C
50 NEXT L
```

P. — Pas mal du tout !... A ce que je vois, tu as dû te creuser la tête pour inventer un tel programme !

O. — Oui, j'avoue !... Mais le résultat en vaut la peine !

P. — Maintenant que ta machine est dessinée, je suppose que tu vas la faire parler puisque tu as prévu une fente pour le texte !



O. — C'est ça !... J'ai envie qu'elle me parle, comme si elle était équipée d'un cerveau artificiel !... Voici d'ailleurs ce que je lui fais dire :

```
60 PRINT AT 3,2;"BONJOUR."
70 PAUSE 50
80 GOSUB 1000
90 PRINT AT 3,2;"JE SUIS UNE M
ACHINE A SOUS."
100 PAUSE 100
110 GOSUB 1000
```

```
120 PRINT AT 3,2;"VOULEZ-VOUS J
OUER AVEC MOI ?"
130 PAUSE 100
140 GOSUB 1000
```

P. — Tu t'amuses bien à ce que je vois !

O. — Oui, pas mal !... On est toutefois prié de ne pas interrompre le programmeur quand il est inspiré !... Je reprends donc là où tu m'as interrompu :

```
150 PRINT AT 3,2;"REPONDEZ PAR
""OUI"" OU ""NON"" "
160 INPUT Z$
170 IF Z$<>"OUI" THEN GOSUB 110
0
```

O. — ... sans oublier les deux sous-programmes 1000 et 1100 !

P. — Dis donc, je vois que tu as prévu large !

O. — N'est-ce pas ?... J'ai comme la vague impression que ce jeu va être long !... Alors, pour éviter de retaper ou de renuméroter des lignes d'instructions, j'ai préféré être prévoyant dès le départ, même si j'ai prévu trop grand !

P. — Tu as bien fait !... Il vaut mieux prévoir trop grand que trop petit !

O. — Bien !... Voici donc les deux sous-programmes en question !... Celui-là, c'est le sous-programme d'effacement :

```
1000 FOR C=2 TO 29
1010 PRINT AT 3,C;" "
1020 NEXT C
1030 RETURN
```

O. — ... et celui-ci, c'est le sous-programme de réponse à un "NON" :

```
1100 GOSUB 1000
1110 PRINT AT 3,2;"AU REVOIR."
1120 PAUSE 50
1130 GOSUB 1000
1140 GOTO 60
```

P. — Tiens, tu ne termines pas ce dernier sous-programme par un RETURN ?

O. — C'est indispensable ?

P. — Non !... Je voulais néanmoins te le faire remarquer !... En fait, si l'on voulait écrire correctement la ligne d'appel de ce sous-programme, il faudrait remplacer l'instruction GOSUB 1100 par GOTO 1100 !... Mais dans notre cas, il n'est pas indispensable de le faire !

O. — Bien, bien !... Je note !

P. — Maintenant que tu as fait connaissance avec la machine, qu'est-ce que tu prévoies comme suite ?

O. — Je vais commencer par mettre des jetons dans la machine. Penses-tu qu'une dizaine soit suffisante ?

P. — Cela me semble très correct !... Et tu fais cela comment ?

O. — Tout bêtement ainsi :

```
180 LET N=10
190 GOSUB 1000
200 PRINT AT 3,2;"VOUS DISPOSEZ
DE ";N;" JETONS."
210 PAUSE 100
220 GOSUB 1000
230 PRINT AT 3,2;"APPUYEZ SUR L
A TOUCHE ""0"".."
240 LET N=N-1
250 IF INKEY$="0" THEN GOTO 270
260 GOTO 250
270 GOSUB 1000
280 PRINT AT 3,2;"C EST PARTI."
```

P. — C'est un vrai roman-feuilleton que tu nous tapes là !... J'ai noté au passage que, dès le début du jeu, chaque joueur avait droit à dix jetons et que chaque fois qu'il appuyait sur la touche [0], la machine encaissait un jeton !

O. — C'est ça !... Ainsi, tu connaîs le nombre de jetons qui te restent avant chaque nouvelle partie !

P. — Très bien !... Et quand tu dis que « c'est parti », qu'est-ce qui est censé être parti ?

O. — La rotation des disques, pardi !... Tu devrais sortir un peu plus, mon petit Papa !... Un « jackpot » comprend au moins trois disques qui entrent en rotation dès qu'on a introduit une pièce dans la machine et qu'on a appuyé sur un levier. Ici, c'est la touche [0] qui simule la manœuvre du levier !

P. — Tu m'en diras tant !... Et comment envisages-tu de simuler la rotation de tes trois disques ?

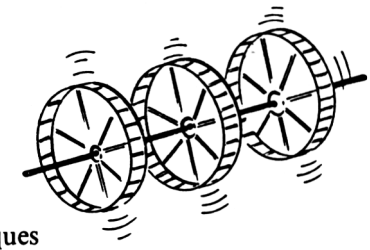
O. — A l'aide du motif hachuré !... Tu vas voir, j'ai déjà essayé, l'effet obtenu n'est pas trop mal !... Voici déjà, pour commencer, la ligne d'appel du sous-programme de rotation des disques :

```
290 GOSUB 1200
```

O. — ... suivie du sous-programme en question :

```
1200 FOR X=2 TO 22 STEP 10
1210 FOR L=7 TO 14
```

```
1220 FOR C=0 TO 7
1230 PRINT AT L,C+X;"▒"
1240 NEXT C
1250 NEXT L
1260 NEXT X
1270 RETURN
```



P. — C'est vrai qu'avec un peu d'imagination on peut arriver à croire que les disques tournent !... Et tu penses les arrêter comment !... Tous ensemble ?

O. — Non !... L'un après l'autre !... Comme ça, le « suspense » sera plus grand !

P. — J'en tremble déjà à l'avance !

O. — Tu peux !... Voici comment je vois la programmation de cette partie « angoissante ».

```
 300 GOSUB 1000
 310 PRINT AT 3,2;"APPUYEZ SUR L
A TOUCHE ""1"".""
 320 IF INKEY$="1" THEN GOTO 340
 330 GOTO 320
 340 LET X=2
 350 GOSUB 1300
```

P. — Je parie que ton GOSUB 1300 correspond à un sous-programme d'arrêt de la rotation du premier disque !... Je me trompe ?

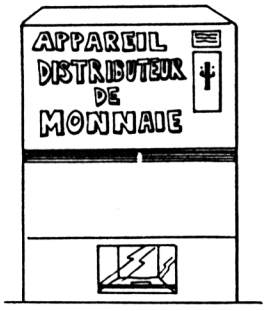
O. — Non, c'est bien ça !... J'ai même pensé me resservir trois fois du même en précisant à la ligne 340 la valeur de la variable X, ce qui, pour le premier disque, est X = 2, pour le second X = 12 et X = 22 pour le troisième.

```
1300 FOR L=7 TO 14
1310 FOR C=0 TO 7
1320 PRINT AT L,C+X;"  "
1330 NEXT C
1340 NEXT L
1350 RETURN
```

P. — Très bien !... Et maintenant, je suppose que tu vas faire apparaître l'un des trois symboles que tu as choisi !

O. — C'est ça !... D'ailleurs, j'ai déjà établi un tableau qui indique combien on gagne de jetons en fonction de la combinaison qui apparaît. Comme tu peux le voir, on gagne 6 jetons avec trois dollars, cinq avec trois livres, etc..., pour terminer avec un seul jeton quand on n'a que deux étoiles !

Combinaison			Gain
$	$	$	6
£	£	£	5
★	★	★	4
▒	$	$	3
▒	£	£	2
▒	★	★	1



P. — Pas mal, pas mal, quoique, à vue de nez, il me semble que le joueur gagne un peu trop souvent avec ta machine !

O. — Comment ça, trop souvent ?

P. — As-tu calculé combien tu pouvais avoir de combinaisons différentes avec trois symboles ?

O. — Non !... Attends, je compte !... Vingt-cinq, vingt-six et vingt-sept !... Il y a vingt-sept combinaisons !

P. — Et sur ces vingt-sept combinaisons, combien en as-tu de gagnantes ?

O. — Six !

P. — Erreur, il y en a douze !... Si tu comptes bien, tu vas te rendre compte que chacune des trois dernières combinaisons de deux symboles peuvent apparaître de trois façons différentes !

O. — Ah, je vois ce que tu veux dire !... Si je prends le cas des deux dollars, ils peuvent apparaître sur le premier et le deuxième disques, ou sur le premier et le troisième, ou enfin sur le deuxième et le troisième !

P. — C'est cela !... Ainsi donc, tu as douze combinaisons gagnantes sur vingt-sept. Calculons maintenant le nombre de jetons qu'elles sont susceptibles de gagner.

O. — Ça, c'est facile !... Il suffit d'additionner le nombre de jetons que gagne chacune des douze combinaisons et on en trouve... trente trois !

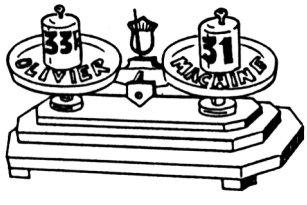
P. — Et rien ne te choque ?

O. — Ben non !

P. — Imagine que chacune des vingt-sept combinaisons sorte une fois. Tu vas donc gagner trente-trois jetons avec vingt-sept. Ce n'est plus une machine à sous mais un distributeur de jetons que tu as fait là !

O. — C'est vrai que c'est un peu trop facile de gagner !... Qu'est-ce qu'on peut faire pour arranger ça ?

P. — Je viens de faire quelques calculs !... Il me semble que si tu rajoutes un carré blanc à tes trois symboles, tu disposeras de soixante-quatre combinaisons au lieu des vingt-sept précédentes, et sur ces soixante-quatre combinaisons, tu auras trente-trois chances de gagner et trente-et-une de perdre !... Cela me paraît plus équitable !



O. — Tu as raison !... Va pour un carré blanc supplémentaire !... Maintenant, comment je vais faire pour les tirer au hasard ?

P. — Comme tu as toujours fait !... Avec des chiffres !... Il te suffit ensuite d'associer un symbole à chaque chiffre !

O. — Bien sûr !... Qu'est-ce que je suis couillon !... Voici donc la suite du programme :

```
360 LET A=INT (RND*4)
370 IF A=1 THEN GOSUB 1400
380 IF A=2 THEN GOSUB 1430
390 IF A=3 THEN GOSUB 1460
```

P. — La variable A va donc correspondre au symbole affiché par le premier disque, j'imagine ?

O. — C'est ça !... Quant aux trois sous-programmes, il s'agit de ceux qui vont dessiner le dollar, la livre et l'étoile !... Tu remarqueras que pour l'espace blanc, qui correspond à A = Ø, il n'y a rien à faire !

SOUS-PROGRAMME DOLLAR

SOUS-PROGRAMME LIVRE

SOUS-PROGRAMME ÉTOILE

P. — J'avais remarqué !... Et comment envisages-tu de dessiner tes symboles ?... A l'aide de chaînes de caractères ?

O. — Non, tout compte fait !... avec les symboles que j'ai choisis, l'économie de mémoire serait nulle !

P. — Ok, je te laisse faire comme tu l'entends !

O. — Voici déjà l'étoile pour commencer :

```
1400 PRINT AT 9,X;"   ■ ■";AT 10,
X;"    ■";AT 11,X;"  ■■■■■■";AT 12,
X;"    ■";AT 13,X;"   ■ ■"
1410 RETURN
```

P. — Je constate que tu as choisi de dessiner la même que celle qui est dans la mémoire des caractères !

O. — Et pour cause !... Je n'ai pas pu faire mieux !... D'ailleurs, j'ai fait pareil pour les deux autres symboles :

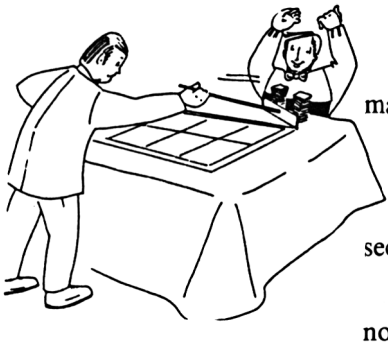
```
1430 PRINT AT 8,X;"    ■■■■";AT 9,
X;"  ■    ■";AT 10,X;" ■■■■■■";AT 1
1,X;"   ■";AT 12,X;"   ■";AT 13,X;
" ■■■■■■■■"
1440 RETURN
1460 PRINT AT 8,X;"    ■";AT 9,X;
" ■■■■■■■";AT 10,X;" ■ ■";AT 11,X;
" ■■■■■■■";AT 12,X;"   ■ ■";AT 13,
X;" ■■■■■■■";AT 14,X;"    ■"
1450 RETURN
```

P. — Voilà une affaire qui s'annonce bien !... On a un dollar qui vient de s'inscrire sur le premier disque. Dépêche-toi donc d'écrire la suite, qu'on sache si l'on a gagné le gros lot !

O. — Ok !... C'est comme si c'était fait !... Voici pour les deux derniers disques :

```
 400 GOSUB 1000
 410 PRINT AT 3,2;"APPUYEZ SUR L
A TOUCHE ""2"".."
 420 IF INKEY$="2" THEN GOTO 440
 430 GOTO 420
 440 LET X=12
 450 GOSUB 1300
 460 LET B=INT (RND*4)
 470 IF B=1 THEN GOSUB 1400
 480 IF B=2 THEN GOSUB 1430
 490 IF B=3 THEN GOSUB 1460
 500 GOSUB 1000
 510 PRINT AT 3,2;"APPUYEZ SUR L
A TOUCHE ""3"".."
 520 IF INKEY$="3" THEN GOTO 540
```

```
530 GOTO 520
540 LET X=22
550 GOSUB 1300
560 LET C=INT (RND*4)
570 IF C=1 THEN GOSUB 1400
580 IF C=2 THEN GOSUB 1430
590 IF C=3 THEN GOSUB 1460
```



P. — J'ai un second dollar et une étoile !

O. — Dans ce cas, tu as gagné 3 jetons !... Bravo pour un début !... A mon tour maintenant !

P. — Et qu'est-ce que tu gagnes ?

O. — Attends !... Voilà une livre pour commencer... un espace blanc... et... une seconde livre !... Youppi, j'ai gagné 2 jetons !

P. — C'est très bien tout cela, mais il faudrait peut-être que ce soit la machine qui nous indique combien de jetons on gagne,... quand on gagne, bien sûr !... Je rajouterai même qu'il faudrait que ladite machine additionne ce gain au nombre de jetons qu'on est censé posséder !

O. — Hou là là, qu'est-ce que tu me demandes là !... Je ne sais pas faire ça, moi !

P. — Tu sembles oublier que l'on est deux dans cette histoire. Je vais te mettre sur la voie !... Additionne donc les valeurs des variables A, B et C et dis-moi à quels totaux tu aboutis !

O. — Ok !... Je trouve donc pour A + B + C les six valeurs suivantes qui correspondent aux combinaisons gagnantes : 9, 6, 3, et 6, 4, 2 !

P. — Ceci fait, il te suffit d'affecter un nombre de jetons donné à chacun de ces six résultats de l'addition !

O. — Oui, mais il y a un problème, car j'ai deux fois le chiffre 6 comme résultat !

P. — Qu'à cela ne tienne !... Si tu réfléchis, tu vas constater que cela provient du choix des valeurs que tu as attribuées au hasard aux variables A, B et C. Essaye avec d'autres valeurs que 1, 2 et 3 !

Et c'est ce qu'il fit !... Je le vis commencer à noicir des feuilles et des feuilles de papier de petits dessins qui ressemblaient à des carrés magiques. De temps en temps j'entendais « En voilà une qui marche ! » entrecoupée fréquemment de « Non, celle-ci n'est pas bonne ! ». Ce n'est que lorsqu'il eut épuisé toutes les combinaisons possibles de 3 nombres de 1 à 9 qu'il me posa cette question :

O. — Dis-donc Papa, j'ai trouvé une douzaine de combinaisons qui conviendraient !... Laquelle tu choisirais, toi ?

P. — Celle que tu n'as pas encore trouvé parce que tu n'as pas cherché assez loin, j'ai nommé 1, 4 et 16 !... En fait, il s'agit des trois premières puissances paires du nombre d'or binaire !... Mais cela, tu ne pouvais le deviner !

O. — Ça ne va pas être facile d'obtenir ces trois nombres avec l'instruction RND !

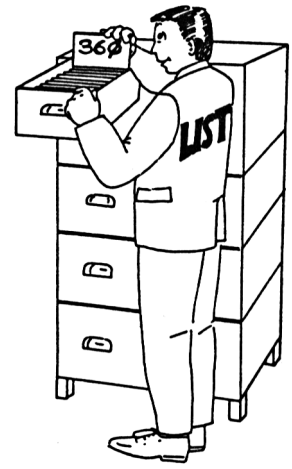
P. — Tel quel, certainement pas !... Par contre, comme il s'agit de carrés parfaits, il suffit d'obtenir les chiffres 1,2 ou 4 qu'on élèvera ensuite au carré et cela, c'est nettement plus facile à réaliser !

O. — Sûr !... Je vais donc modifier en conséquence le programme à partir de la ligne 360 !... Le temps de remonter mon curseur jusque-là !

P. — Oh, mais pour cela, il faut employer l'instruction LIST !... Je me rends compte que j'ai oublier de t'en parler !

O. — Et on l'utilise comment, cette instruction LIST ?

P. — En la faisant suivre du numéro de la ligne devant laquelle tu veux placer le curseur !... Dans ton cas, tu n'as qu'à taper LIST 360 !



O. — Effectivement, qu'est-ce que ça gagne comme temps !... Bon, alors, en avant pour les corrections :

```
360 LET A=INT (RND*5)
365 IF A=3 THEN GOTO 360
390 IF A=4 THEN GOSUB 1460
460 LET B=INT (RND*5)
465 IF B=3 THEN GOTO 460
490 IF B=4 THEN GOSUB 1460
560 LET C=INT (RND*5)
565 IF C=3 THEN GOTO 560
590 IF C=4 THEN GOSUB 1460
```

P. — C'est bien !... Avec ceci, tu peux maintenant aborder la suite, c'est-à-dire le calcul du gain. Rappelle-moi quelles sont les combinaisons gagnantes avec ces nouvelles valeurs de A, B et C ?

O. — 1, 2, 3... 4, 8, 12... et... 16, 32, 48 !... On va d'ailleurs retrouver ces valeurs dans ce que je vais écrire :

```
600 LET S=A**2+B**2+C**2
610 LET G=0
620 IF S=2 OR S=6 OR S=18 THEN
LET G=1
630 IF S=8 OR S=9 OR S=24 THEN
LET G=2
640 IF S=32 OR S=33 OR S=36 THE
N LET G=3
650 IF S=3 THEN LET G=4
660 IF S=12 THEN LET G=5
670 IF S=48 THEN LET G=6
```

P. — C'est pas mal du tout !... Je note que tu as pensé, pour les combinaisons de deux étoiles, de deux livres ou de deux dollars, à tenir compte de la valeur du troisième symbole !

O. — Je vais t'avouer que je n'y ai pas pensé tout de suite !... C'est seulement quand j'ai réalisé qu'on faisait l'addition des trois variables que je me suis rendu compte qu'il fallait tenir compte des trois autres symboles qui pouvaient accompagner une paire gagnante !

P. — Bravo !... Il ne nous reste plus, pour terminer ce jeu, qu'à indiquer si on a gagné ou si on a perdu, et à remettre en jeu automatiquement !

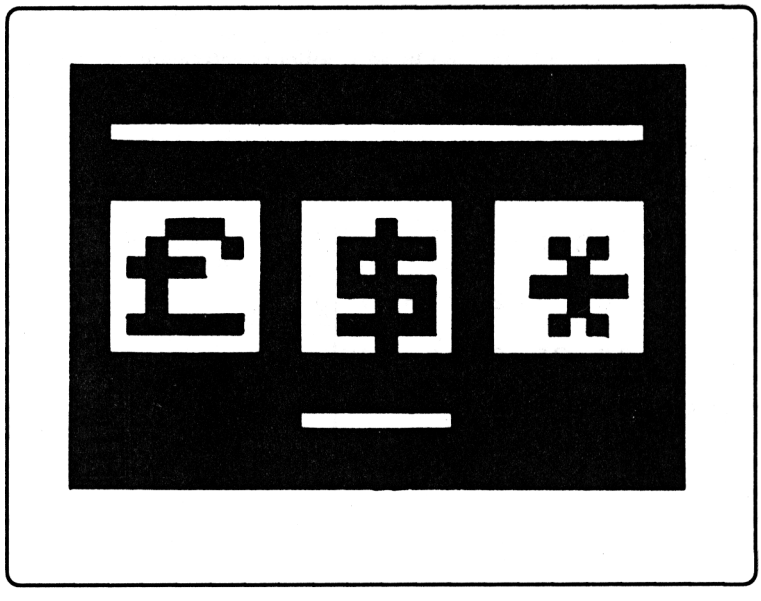
O. — Tout ça me paraît assez simple !... Voici déjà pour l'affichage des résultats :

```
680 GOSUB 1000
690 IF G=0 THEN PRINT AT 3,2;"V
OUS AVEZ PERDU."
700 IF G>0 THEN GOSUB 1500
```

O. — ... suivi du sous-programme 1500 qui indique que l'on a gagné et qui dessine les jetons qui sortent de la machine :

```
1500 PRINT AT 3,2;"VOUS AVEZ GAG
NE ";G;" JETONS."
1510 FOR J=1 TO G
1520 PRINT AT 18,12+J;"O"
1530 NEXT J
1540 PAUSE 200
1550 FOR J=1 TO G
1560 PRINT AT 18,12+J;" "
1570 NEXT J
1580 RETURN
```

P. — L'effet obtenu est très bien !... On croirait entendre les jetons tomber !... Chapeau !



O. — Je continue !... Tu vas voir, c'est drôle :

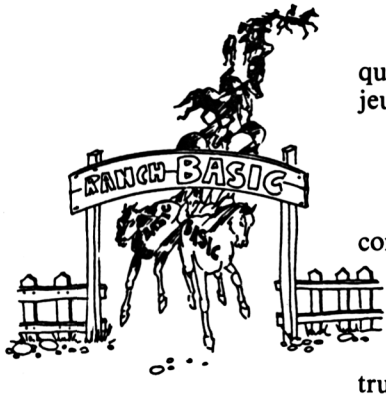
```
710 GOSUB 1000
720 LET N=N+G
730 IF N=0 THEN GOTO 760
740 IF N=99 THEN GOTO 780
750 GOTO 200
760 PRINT AT 3,2;"IL NE VOUS RE
STE PLUS RIEN."
```

```
770 STOP
780 PRINT AT 3,2;"DESOLE, LA CA
ISSE EST VIDE."
```

P. — Effectivement, je trouve cela très drôle que tu aies pensé au joueur tellement chanceux qu'il risque de vider la caisse de la machine !... Bravo pour l'imagination !

Mais Olivier ne m'écoutait déjà plus !... Il était en train de jouer avec le jeu vidéo qu'il avait inventé et, dans le moment présent, c'était pour lui la plus belle chose qui puisse exister au monde !

LES INSTRUCTIONS EN BASIC



Le dernier jeu que nous venions de réaliser ensemble, Olivier et moi, terminait en quelque sorte l'expérience d'apprentissage du langage BASIC à travers la réalisation de jeux vidéo. Je le questionnai justement à ce sujet là :

P. — Alors, le BASIC, qu'est-ce tu en penses ?

O. — Le BASIC ?... Mais c'est très simple !... Je ne me suis même pas rendu compte du nombre d'instructions que j'ai pu apprendre !... Tu en as fait le compte, toi ?

P. — Oui !... En tout et pour tout, nous en avons employé 45 !

O. — Ben dis donc !... Ça en fait une quantité !... Est-ce qu'on a vu toutes les instructions du BASIC ?

P. — Non, pas tout-à-fait !... Il en reste encore à peu près une douzaine que j'ai passé sous silence !... Je te conseille quand tu auras un moment de jeter un coup d'œil à ton manuel d'utilisation à leur sujet !... Tu verras que là encore, c'est tout simple !

O. — Au sujet des instructions qu'on a employées, tu sais ce qu'il faudrait faire ?

P. — Dis toujours !

O. — Il faudrait établir un tableau dans lequel figurerait chaque instruction et où serait indiqué la page de ton livre où elle est décrite pour la première fois !

P. — Très bonne ton idée !... Ainsi, si tu veux te rappeler à quoi sert une instruction et quel en est le mode d'emploi, il te suffit de te reporter à ce tableau !... Je le dresse tout de suite :

O. — Super !... Voilà un tableau qui va être drôlement utile !

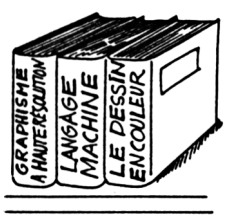
P. — Il me reste encore une question que j'aimerais bien te poser !

O. — Vas-y !

P. — As-tu aimé les jeux vidéo que nous avons réalisés ensemble ?

O. — Oui et non !... Oui, parce que ce sont les jeux que je voulais faire et non, parce que le résultat obtenu n'est pas aussi bon que ceux qu'on peut acheter sous forme de cassettes !

P. — C'est normal !... Ne crois pas que tu sais tout sur les jeux vidéo et l'informatique !... En fait, tu n'as progressé que d'une marche sur un escalier qui en comporte des dizaines !... Pour ne te citer que quelques exemples, je suppose que tu aimerais bien pouvoir accélérer le déplacement de tes engins dans certains jeux ?



O. — Ça oui !

P. — Eh bien pour cela, il faut faire appel au langage machine et cela, c'est une autre marche à escalader !

Les instructions en BASIC

Instruction BASIC	Page	Instruction BASIC	Page	Instruction BASIC	Page
RUBOUT	8	LET	38	BREAK	100
PRINT	9	CHR$	49	SGN	108
NEWLINE	9	PLOT	51	DIM	115
AT	11	AND	58	UNPLOT	117
FUNCTION	12	OR	58	CODE	125
GRAPHICS	14	GOSUB	59	TAN	130
RUN	18	RETURN	59	INKEY$	135
NEW	19	SPACE	63	NOT	161
FOR	19	GOTO	70	FAST	178
TO	19	RND	77	SLOW	178
NEXT	19	INT	78	PAUSE	188
EDIT	23	SIN	84	PEEK	188
STEP	37	COS	84	STOP	198
IF	38	SCROLL	87	INPUT	240
THEN	38	CLS	98	LIST	251

O. — Ah bon !

P. — Je suppose aussi que tu aimerais pouvoir dessiner sur l'écran avec une plus grande finesse que celle qu'autorise les petits carrés !... Je me trompe ?

O. — Non !... C'est ça qui serait super chouette !

P. — Eh bien pour cela, il te faut étudier les possibilités du graphisme à haute résolution qu'offrent certaines machines !

O. — Encore une marche ?

P. — C'est bien cela !... Et la couleur, cela ne te tenterait pas de dessiner en couleur ?

O. — Tu penses bien, c'est mon rêve !

P. — Dans ce cas, il te faut employer un micro-ordinateur plus évolué qui offre cette possibilité !... Encore une marche !

O. — Arrête !... Toutes ces marches me donnent le vertige !... Tu ne voudrais pas m'aider à les monter une à une ?

P. — Je veux bien, mais ce que tu me demandes là, c'est une autre histoire !

TABLE DES MATIÈRES

Comment dessiner sur un écran 7
La mise en route du micro-ordinateur — Le quadrillage de l'écran — La fonction graphique — Comment tracer une ligne droite — Le mode programme — Les horizontales — Les verticales — Les obliques — Les pavés.

Les aires de jeux 31
Le pourtour d'un terrain — Le tracé continu — Un terrain de football — Une piscine de compétition — Une table de ping-pong — Un quadrillage de bataille navale — L'affichage des coordonnées — Un damier d'échecs — Un court de tennis — Un labyrinthe.

Engins et décors 65
Une soucoupe volante — Quelques véhicules extraterrestres — La navette spatiale et des avions — Un train, une voiture de course et un camion — Un sous-marin et des bateaux — Animaux et personnages — Un parcours de golf — Une caverne — Une ville — Une base lunaire — Une route — Une piste de ski — Une ceinture d'astéroïdes.

Animons les images 93
Déplacement de gauche à droite — Déplacement de droite à gauche — Apparition et disparition progressives sur l'écran — Déplacements en sens inverse — Les déplacements verticaux — Comment modifier la vitesse — Les variables indicées — Les déplacements en oblique — La trajectoire balistique.

Aux commandes du clavier 133
Les commandes de montée et de descente — La commande de déplacement latéral — La commande de changement de vitesse — La commande de largage — La commande de tir — Le tir laser à partir d'un véhicule en déplacement.

Rebonds et impacts 167
Les rebonds à l'horizontale — Les rebonds à la verticale — Les rebonds en oblique — Le rebond sur une raquette — Eclatement d'un astéroïde — Un rendez-vous spatial — L'instruction PEEK — Obstacles sur une route — Déplacement dans un labyrinthe.

La touche finale 205
L'affichage des résultats — La remise en jeu automatique — Les chaînes de caractères — Le chronométrage du temps — Les gros chiffres — Les cartes à jouer.

Les instructions en BASIC 254

Table des matières 256

Service lecteurs 257

Comment créer et animer ses jeux vidéo
1re édition !

S.E.C.F.

EDITIONS RADIO

Service lecteurs

(à retourner à S.E.C.F.-Éditions Radio, 9, rue Jacob, 75006 Paris)

Pour nous permettre de vous proposer des ouvrages toujours meilleurs, nous souhaiterions recevoir vos critiques, appréciations et suggestions sur le présent livre :

_____

_____

_____

_____

Quels sont les ouvrages (thème, sujet, niveau) que vous souhaiteriez voir publier par notre société ?

_____

_____

_____

_____

Nous vous remercions de votre confiance et de votre coopération.

S.E.C.F.-Éditions Radio

Je désire recevoir gratuitement et sans engagement (mettre une croix dans la case) :

☐ Votre catalogue général (Electronique professionnelle et grand public, Informatique, Hi-Fi, Vidéo)

☐ Votre catalogue spécial informatique.

Nom : _____ Prénom : _____

Adresse : _____

_____

Secteur d'activité et fonction : _____

_____

CENTRES D'INTÉRÊTS

☐ Electronique professionnelle

☐ Electronique de loisirs

☐ Vidéo

☐ Hifi, CB...

☐ Micro-informatique professionnelle

☐ Micro-informatique de loisirs

☐ Autres : .....

.....

Voir au dos "Correspondance Auteurs"

Comment créer et animer ses jeux vidéo
1re édition

S.E.C.F.

EDITIONS RADIO

Service lecteurs

Correspondance auteurs

(à retourner à S.E.C.F.-Éditions Radio, 9, rue Jacob, 75006 Paris)

Pour toute demande d'éclaircissements techniques relatifs à ce livre, formulez ci-dessous vos questions, avec le maximum de précisions. Nous les transmettrons à l'auteur qui ne manquera pas de vous répondre directement :

Nom : _____ Prénom : _____

Adresse : _____

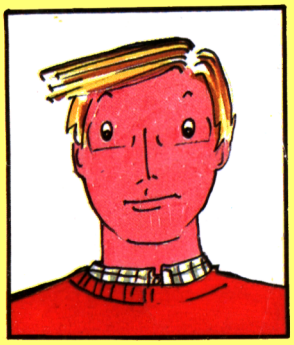
S. E. C. F.

Editions Radio

9, RUE JACOB - 75006 PARIS
TEL. 329.63.70

sur ZX 81

CRÉER ANIMER ses JEUX VIDÉO



P - Alors Olivier, qu'est-ce que tu penses de ce livre ?

O - Il est franchement super !

P - Et les jeux, comment les trouves-tu ?

O - Pas mal, bien sûr, mais ils ne valent pas le dernier que je viens d'inventer tout seul !

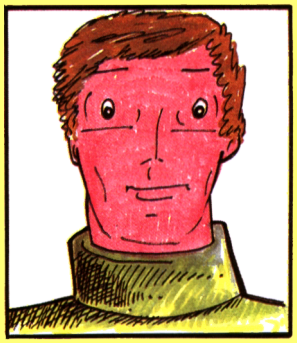
P - Et le BASIC dans tout ça ?

O - Le BASIC ?... Alors là, je dois avouer que je ne me suis même pas rendu compte que je l'apprenais en construisant tous ces jeux !

P - Et l'avenir, pour toi, c'est quoi ?

O - C'est travailler sur un ordinateur plus puissant et apprendre le langage machine, le graphisme à haute résolution et peut-être aussi le PASCAL.

P - Quel programme...



S. E. C. F.

ÉDITIONS RADIO

9 782709 109376

ISBN 2 7091 0937 9

Code 84

Prix : 100 F

CRÉER ANIMER SES JEUX VIDÉO

J.C. FANTOU

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/

[FRA] Ce document a été préservé numériquement à des fins éducatives et d'études, non commerciales.
[ENG] This document has been digitally preserved for educational and study purposes, not for commercial purposes.
[ESP] Este documento se ha conservado digitalmente con fines educativos y de estudio, no con fines comerciales.